FUJITSU SEMICONDUCTOR

CONTROLLER MANUAL

CM44-10136-1E

F^{2}MC^{TM}-16LX

16-BIT MICROCONTROLLER

MB90360 Series

HARDWARE MANUAL

F^2MC^{TM}-16LX

16-BIT MICROCONTROLLER

MB90360 Series

HARDWARE MANUAL

FUJITSU LIMITED

PREFACE

■ Objectives and intended reader

Thank you very much for your continued patronage of Fujitsu semiconductor products.

The MB90360 series has been developed as a general-purpose version of the F^{2}MC-16LX family, which is an original 16-bit single-chip microcontroller compatible with the Application Specific IC (ASIC).

This manual explains the functions and operation of the MB90360 series for engineers who actually use the MB90360 series to design products. Please read this manual first.

■ Trademark

F^{2}MC, an abbreviation of FUJITSU Flexible Microcontroller, is a registered trademark of FUJITSU Ltd.

Embedded Algorithm is a registered trademark of Advanced Micro Devices Inc.

■ Structure of this preliminary manual

This manual contains the following 26 chapters and appendix.

CHAPTER 1 OVERVIEW

The MB90360 Series is a family member of the F^{2}MC-16LX micro controllers.

CHAPTER 2 CPU

This chapter explains the CPU.

CHAPTER 3 INTERRUPTS

This chapter explains the interrupts and function and operation of the extended intelligent I/O service in the MB90360 series.

CHAPTER 4 DELAYED INTERRUPT GENERATION MODULE

This chapter explains the functions and operations of the delayed interrupt generation module.

CHAPTER 5 CLOCKS

This chapter explains the clocks used by MB90360 series microcontrollers.

CHAPTER 6 CLOCK SUPERVISOR

This chapter explains the function and the operation of the clock supervisor. Only the product with built-in clock supervisor of the MB90360 series is valid to this function.

CHAPTER 7 RESETS

This chapter describes resets for the MB90360-series microcontrollers.

CHAPTER 8 LOW-POWER CONSUMPTION MODE

This chapter explains the low-power consumption mode of MB90360 series microcontrollers.

CHAPTER 9 MEMORY ACCESS MODES

This chapter explains the functions and operations of the memory access modes.

CHAPTER 10 I/O PORTS

This chapter explains the functions and operations of the I/O ports.

CHAPTER 11 TIMEBASE TIMER

This chapter explains the functions and operations of the timebase timer.

CHAPTER 12 WATCHDOG TIMER

This chapter describes the function and operation of the watchdog timer.

CHAPTER 13 16-Bit I/O TIMER

This chapter explains the function and operation of the 16- bit I/O timer.

CHAPTER 14 16-BIT RELOAD TIMER

This chapter describes the functions and operation of the 16-bit reload timer.

CHAPTER 15 WATCH TIMER

This chapter describes the functions and operations of the watch timer.

CHAPTER 16 8-/16-BIT PPG TIMER

This chapter describes the functions and operations of the 8-/16-bit PPG timer.

CHAPTER 17 DTP/EXTERNAL INTERRUPTS

This chapter explains the functions and operations of DTP/external interrupt.

CHAPTER 18 8-/10-BIT A/D CONVERTER

This chapter explains the functions and operation of 8-/10-bit A/D converter.

CHAPTER 19 LOW VOLTAGE DETECTION/CPU OPERATING DETECTION RESET

This chapter explains the function and operating the low voltage detection/CPU operating detection reset. This function can use only the product with "T" suffix of MB90360 series.

CHAPTER 20 LIN-UART

This chapter explains the functions and operation of LIN-UART.

CHAPTER 21 CAN CONTROLLER

This chapter explains the functions and operations of the CAN controller.

CHAPTER 22 ADDRESS MATCH DETECTION FUNCTION

This chapter explains the address match detection function and its operation.

CHAPTER 23 ROM MIRRORING MODULE

This chapter describes the functions and operations of the ROM mirroring function select module.

CHAPTER 24 512K-BIT FLASH MEMORY

This chapter explains the functions and operation of the 512K-bit flash memory. The following three methods are available for writing data to and erasing data from the flash memory:

- Parallel programmer
- Serial programmer
- Executing programs to write/erase data

This chapter explains “Executing programs to write/erase data”.

CHAPTER 25 EXAMPLES OF MB90F362/T(S), MB90F367/T(S) SERIAL PROGRAMMING CONNECTION

This chapter shows an example of a serial programming connection using the AF220/AF210/AF120/AF110 Flash Micro-computer Programmer by Yokogawa Digital Computer Corporation when the AF220/AF210/AF120/AF110 flash serial microcontroller programer from Yokogawa Digital Computer Corporation is used.

CHAPTER 26 ROM SECURITY FUNCTION

This chapter explains the ROM security function.

APPENDIX

The appendixes provide I/O maps, instructions, and other information.

- The contents of this document are subject to change without notice.
  Customers are advised to consult with FUJITSU sales representatives before ordering.
- The information, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of Fujitsu semiconductor device; Fujitsu does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. Fujitsu assumes no liability for any damages whatsoever arising out of the use of the information.
- Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of Fujitsu or any third party or does Fujitsu warrant non-infringement of any third-party' s intellectual property right or other right by using such information. Fujitsu assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.
- The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).
  Please note that Fujitsu will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.
- Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
- If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the prior authorization by Japanese government will be required for export of those products from Japan.

CONTENTS

CHAPTER 1
OVERVIEW

The MB90360 Series is a family member of the F^{2}MC-16LX micro controllers.

1.1 Overview of MB90360

The MB90360 Series is a 16-bit microcontroller designed for automotive applications and contains CAN function, capture, compare timer, A/D converter, and so on.

■ Features of MB90360 Series

MB90360 series has the following features:

● Clock

- Built-in PLL clock multiplying circuit
- Machine clock (PLL clock) selectable from 1/2 frequency of oscillation clock or 1 to 6-multiplied oscillation clock (4 MHz to 24 MHz when oscillation clock is 4 MHz)
- Subclock operation (8.192 kHz)
- Minimum instruction execution time: 42 ns (4-MHz oscillation clock and 6-multiplied PLL clock)
- Clock supervisor: monitors main clock or subclock independently
- Subclock mode: Clock source selectable from external oscillator or internal CR oscillator

● 16-MB CPU memory space

- Internal 24-bit addressing

● Instruction system optimized for controllers

- Various data types (bit, byte, word, long word)
- 23 types of addressing modes
- Enhanced signed instructions of multiplication/division and RETI
- High-accuracy operations enhanced by 32-bit accumulator

● Instruction system for high-level language (C language)/multi-task

- System stack pointer
- Enhanced pointer indirect instructions
- Barrel shift instructions

● Higher execution speed

4 bytes instruction queue

● Powerful interrupt function

- Powerful interrupt function with 8 levels and 34 factors
- Corresponds to 8-channel external interrupts

● CPU-independent automatic data transfer function

Extended intelligent I/O service (EI^{2}OS): Maximum 16 channels

● Lower-power consumption (standby) modes

- Sleep mode (stops CPU clock)
- Timebase timer mode (operates only oscillation clock and subclock, timebase timer and watch timer)
- Watch mode (product without S-suffix operates only subclock and watch timer)
- Stop mode (stops oscillation clock and subclock)
- CPU intermittent operation mode

● Process

CMOS Technology

● I/O ports

- General-purpose I/O ports (CMOS output)
  - 34 ports (product without S-suffix)
  - 36 ports (product with S-suffix)

● Subclock pin (X0A, X1A)

- Yes (external oscillator used) ... products without S-suffix
- No (subclock mode is used with internal CR oscillation) ... product with S-suffix

● Timers

- Timebase timer, watch timer (product without S-suffix), watchdog timer: 1 channel
- 8-/16-bit PPG timer: 8 bits × 4 channels or 16 bits × 2 channels
- 16-bit reload timer: 2 channels
- 16-bit I/O timer
  - 16-bit free-run timer: 1 channel (FRT0: ICU0/1/2/3)
  - 16-bit input capture (ICU): 4 channels

● Full-CAN* CAN Controller: 1 channel

- Conforms to CAN Specification Ver. 2.0A and Ver. 2.0B.
- Built-in 16 message buffers
- CAN wake up

● UART (LIN/SCI): Maximum 2 channels

- Full-duplex double buffer
- Clock asynchronous or clock synchronous serial transfer

● DTP/external interrupt: 8 channels, CAN wake up: 1 channel

External input to start EI^{2}OS and generate external interrupt

- Delayed interrupt generation module

  Generates interrupt request for task switching

- 8-/10-bit A/D converter: 16 channels

  - 8-bit and 10-bit resolutions
  - Start by external trigger input
  - Conversion time: 3 µs (including sampling time at 24-MHz machine clock frequency)

- Program patch function

  Detects address match for six address pointers

- Low voltage/CPU operation detection reset function (product with T-suffix)

  - Detects low voltage (4.0 V ± 0.3 V) and reset automatically
  - Automatic reset when program runs away and counter is not cleared within internal time (approx. 262 ms @ 4 MHz external)

- Clock supervisor (MB90x367x only)

- Changeable port input voltage level

  Automotive input level/CMOS Schmitt input level (initial value in single-chip mode is Automotive level)

- ROM security function

  Capable of protecting the content of ROM (MASK ROM product only)

*: Controller Area Network (CAN) - License of Robert Bosch GmbH

■ Product overview

Table 1.1-1 Product Overview (1/2)

Features	MB90362	MB90362T	MB90362S	MB90362TS	MB90V340A-101	MB90V340A-102
CPU	F^2MC-16LX CPU					
System clock pin	PLL clock multiplier (×1, ×2, ×3, ×4, ×6, 1/2 when PLL stops) Minimum instruction execution time: 42 ns (4 MHz osc. PLL ×6)					
Sub clock pin (X0A, X1A)	Yes		No		No	Yes
Clock supervisor	No					
ROM	MASK ROM, 64K bytes				External	
RAM capacitance	3K bytes				30K bytes	
CAN interface	1 channel				3 channels	
Low voltage/CPU operation detection reset	No	Yes	No	Yes	No	
Package	LQFP-48				PGA-299	
Power supply for emulator*	-				Yes	
Corresponding EVA product name	MB90V340A-102		MB90V340A-101			

*: It is setting of Jumper switch (TOOL V_{CC}) when Emulator (MB2147-01) is used. Please refer to Emulator hardware manual.

Features	MB90F362	MB90F362T	MB90F362S	MB90F362TS
CPU	F^2MC-16LX CPU			
System clock pin	PLL clock multiplier (×1, ×2, ×3, ×4, ×6, 1/2 when PLL stops) Minimum instruction execution time: 42 ns (4 MHz osc. PLL ×6)			
Sub clock pin (X0A, X1A)	Yes		No	
Clock supervisor	No			
ROM	Flash memory, 64K bytes			
RAM capacitance	3K bytes			
CAN interface	1 channel			
Low voltage/CPU operation detection reset	No	Yes	No	Yes
Package	LQFP-48			
Corresponding EVA product name	MB90V340A-102		MB90V340A-101	

Table 1.1-2 Product Overview (2/2)

Features	MB90367	MB90367T	MB90367S	MB90367TS	MB90V340 A-103	MB90V340 A-104
CPU	F^{2}MC-16LX CPU					
System clock pin	PLL clock multiplier (×1, ×2, ×3, ×4, ×6, 1/2 when PLL stops) Minimum instruction execution time: 42 ns (4 MHz osc. PLL ×6)					
Sub clock pin (X0A, X1A)	Yes		No (Internal CR oscillation can be used as subclock)			Yes
Clock supervisor	Yes					
ROM	MASK ROM, 64K bytes				External	
RAM capacitance	3K bytes				30K bytes	
CAN interface	1 channel				3 channels	
Low voltage/CPU operation detection reset	No	Yes	No	Yes	No	
Package	LQFP-48				PGA-299	
Power supply for emulator*	-				Yes	
Corresponding EVA product name	MB90V340A-104		MB90V340A-103			

*: It is setting of Jumper switch (TOOL V$_{CC}$) when Emulator (MB2147-01) is used. Please refer to Emulator hardware manual.

Features	MB90F367	MB90F367T	MB90F367S	MB90F367TS
CPU	F^{2}MC-16LX CPU			
System clock pin	PLL clock multiplier (×1, ×2, ×3, ×4, ×6, 1/2 when PLL stops) Minimum instruction execution time: 42 ns (4 MHz osc. PLL ×6)			
Sub clock pin (X0A, X1A)	Yes		No (Internal CR oscillation can be used as subclock)	
Clock supervisor	Yes			
ROM	Flash memory, 64K bytes			
RAM capacitance	3K bytes			
CAN interface	1 channel			
Low voltage/CPU operation detection reset	No	Yes	No	Yes
Package	LQFP-48			
Corresponding EVA product name	MB90V340A-104		MB90V340A-103	

■ Features

Table 1.1-3 MB90360 Features (1/2)

Features	MB90F362/T(S), MB90362/T(S) MB90F367/T(S), MB90367/T(S)	MB90V340A-101, MB90V340A-102 MB90V340A-103, MB90V340A-104
UART	2 channels	5 channels
	Wide range of baud rate settings using a dedicated reload timer LIN functionality working either as LIN master or LIN slave device	
A/D converter	16 channels	24 channels
	10-bit or 8-bit resolution Conversion time: Minimum 3 µs include sample time (per one channel)	
16-bit reload timer	2 channels	4 channels
	Operation clock frequency: fsys/2^1, fsys/2^3, fsys/2^5(fsys=System clock freq.) Support External Event Count function	
16-bit I/O timer	1 channel	4 channels
	I/O timer 0 (clock input FRCK0) corresponding ICU 0/1/2/3.	I/O timer 0 (clock input FRCK0) corresponds to ICU 0/1/2/3, OCU 0/1/2/3 I/O timer 1 (clock input FRCK1) corresponds to ICU 4/5/6/7, OCU 4/5/6/7
	Signal an interrupt when overflowing Supports Timer Clear when a match with Output Compare (Channel 0, 4) Operation clock freq.: fsys/2^1, fsys/2^2, fsys/2^3, fsys/2^4, fsys/2^5, fsys/2^6, fsys/2^7 (fsys=System clock freq.)	
16-bit input capture	4 channels	8 channels
	Maintains I/O timer value by pin input (rising edge, falling edge, or both edges) and generates interrupt.	
8-/16-bit PPG	2 channels	8 channels
	Supports 8-bit and 16-bit operation modes Four 8-bit reload counter Four 8-bit reload registers for L pulse width Four 8-bit reload registers for H pulse width	Supports 8-bit and 16-bit operation modes Sixteen 8-bit reload counter Sixteen 8-bit reload registers for L pulse width Sixteen 8-bit reload registers for H pulse width
	A pair of 8-bit reload counters can be configured as one 16-bit reload counter or as 8-bit prescaler plus 8-bit reload counter Operation clock freq.: fsys, fsys/2^1, fsys/2^2, fsys/2^3, fsys/2^4 or 102.4 µs fosc=@5MHz (fsys=system clock frequency, fosc=oscillation clock frequency)	

Table 1.1-3 MB90360 Features (2/2)

Features	MB90F362/T(S), MB90362/T(S) MB90F367/T(S), MB90367/T(S)	MB90V340A-101, MB90V340A-102 MB90V340A-103, MB90V340A-104
CAN interface	1 channel	3 channels
	Conforms to CAN Specification Version 2.0 Part A and B Automatic re-transmission in case of error Automatic transmission responding to Remote Frame 16 message buffers for transmission/reception Supports multiple messages Flexible configuration of acceptance filtering: • Full bit compare/Full bit mask/2 partial bit masks • Supports up to 1 Mbps communication	
External interrupt (8 channels)	Can be programmed edge sensitive or level sensitive	
D/A converter	-	2 channels
Low voltage/CPU operation detection reset	Corresponds to product with T-suffix only	-
Clock supervisor	MB90F367/T(S), MB90367/T(S) only	-
Subclock (Maximum 100 kHz)	Corresponds to product without T-suffix only Corresponds to MB90V340A-102/MB90V340A-104 only	
I/O port	Supports general-purpose I/O (CMOS output): - 34 ports (product without S-suffix) - 36 ports (product with S-suffix) Input level setting: - Port2, Port4, Port6, Port8: selectable from CMOS/Automotive level	Supports general-purpose I/O (CMOS output): - 80 ports (product without S-suffix) - 82 ports (product with S-suffix) Input level setting - Port 0 to Port 3: selectable from CMOS/Automotive/TTL level - Port 4 to Port A: selectable from CMOS/Automotive level
Flash memory	Supports automatic programming, Embedded Algorithm^{TM*}, Write/Erase/Erase-Suspend/Resume commands A flag indicating completion of the algorithm Number of erase cycles: 10,000 times Data retention time: 20 years Flash Security Feature for protecting the content of the Flash	
ROM security	Protects the content of ROM (MASK ROM product only)	-

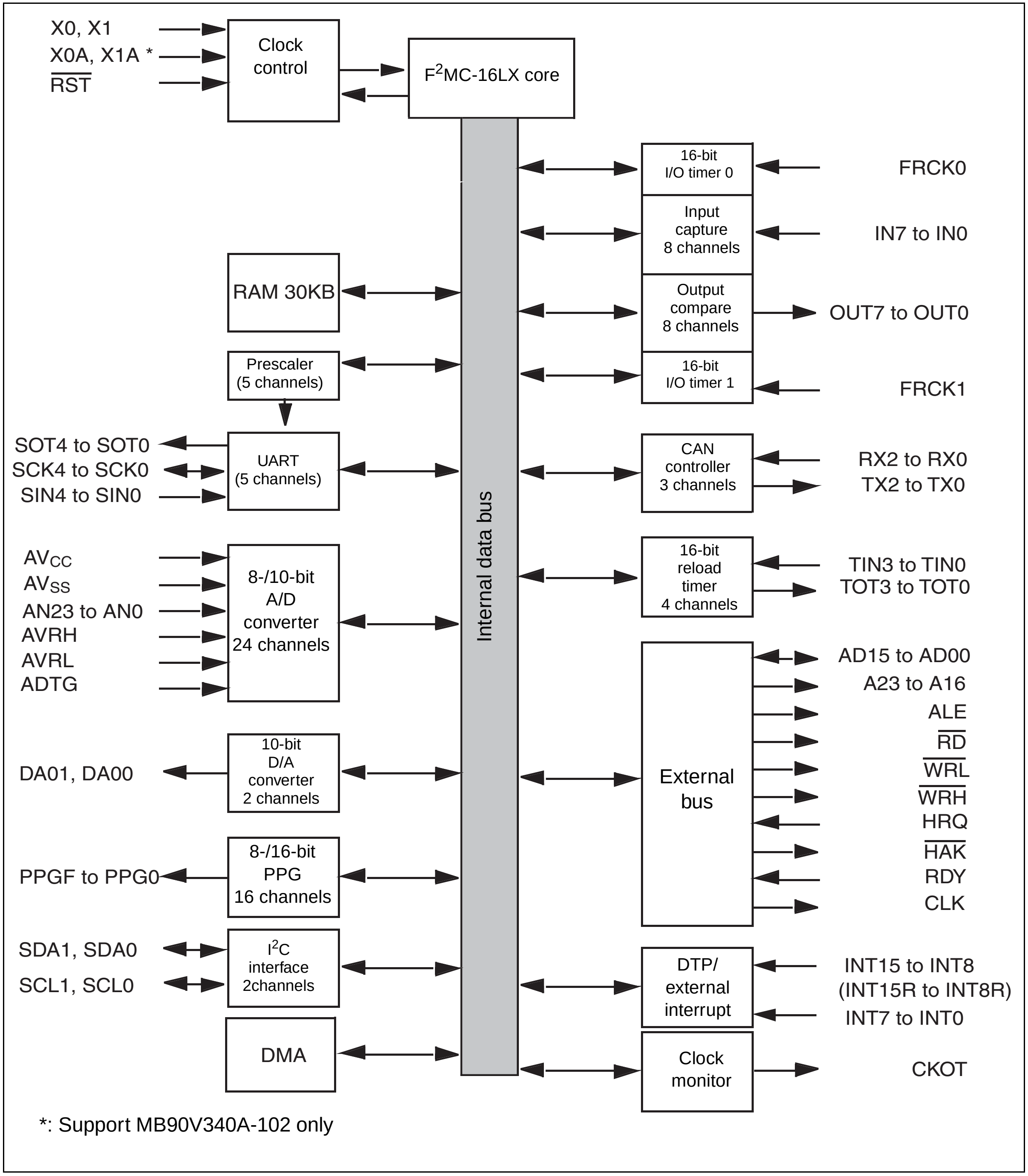
*:Embedded Algorithm is a registered trademark of Advanced Micro Device Inc.

1.2 Block Diagram of MB90360 series

Figure 1.2-3 shows a block diagram of the MB90360.

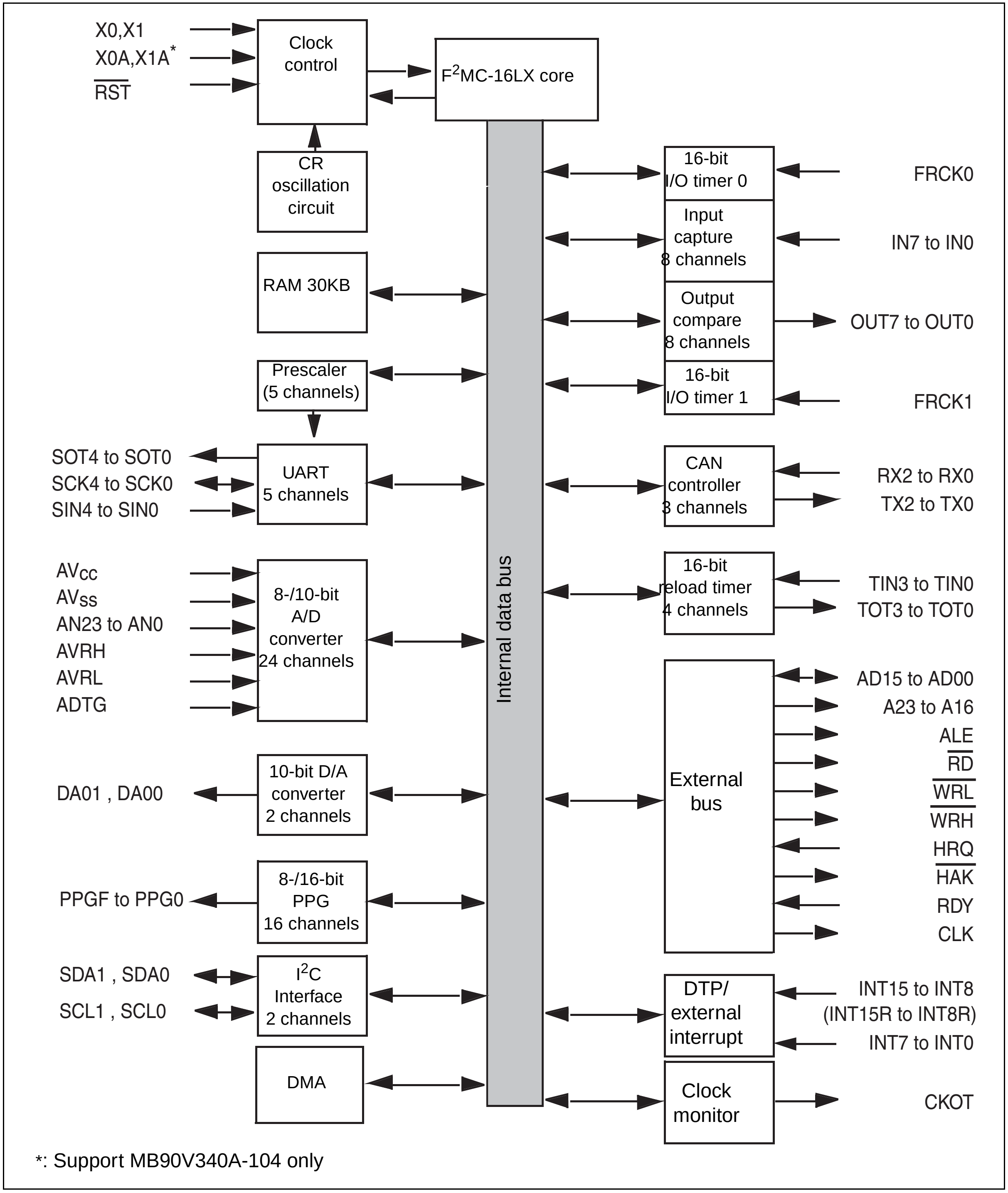
■ Block Diagram of Evaluation Chip

Figure 1.2-1 Block Diagram of Evaluation Chip (MB90V340A-101/102)



*: Support MB90V340A-102 only

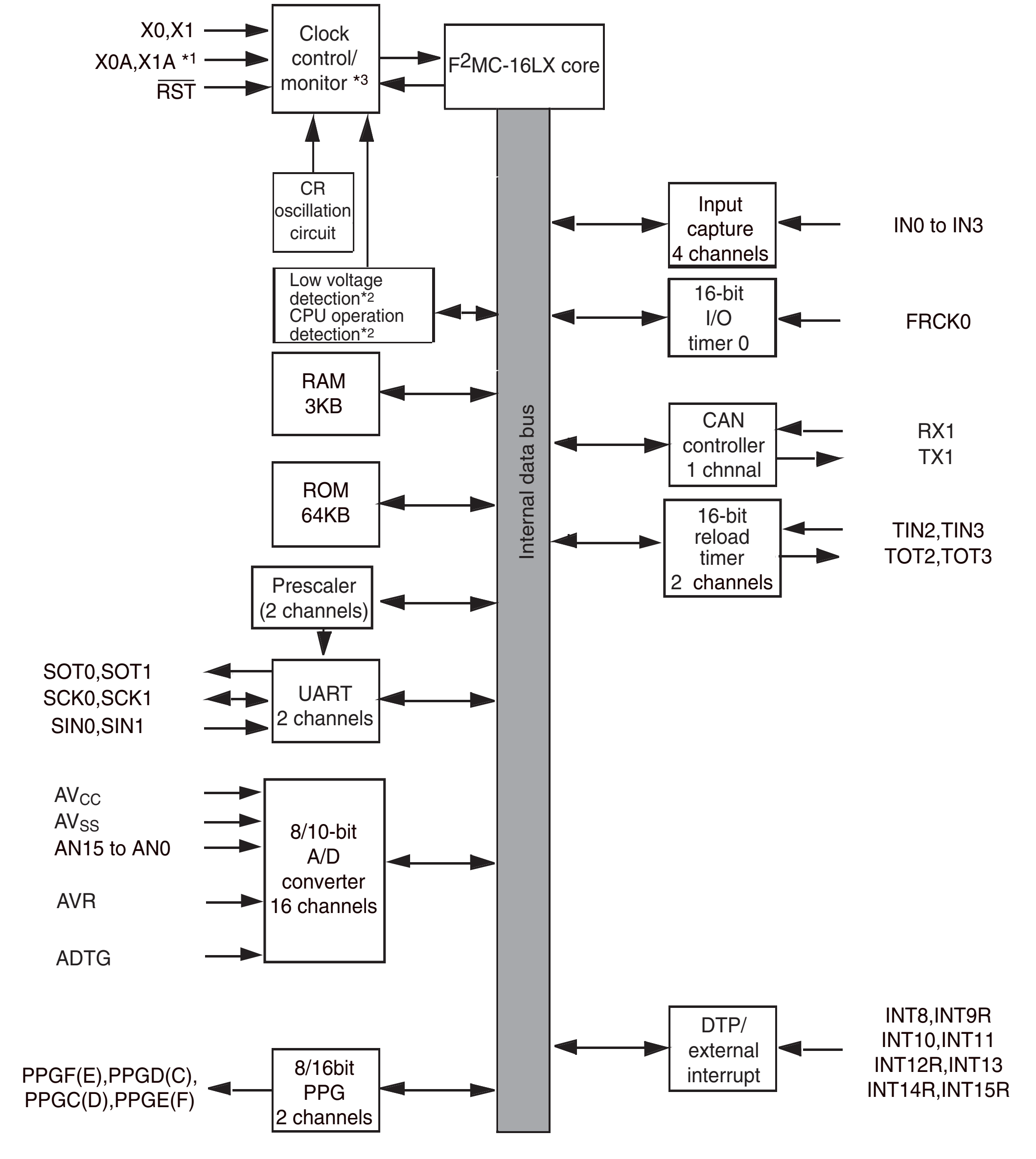
Figure 1.2-2 Block Diagram of Evaluation Chip (MB90V340A-103/104)



*: Support MB90V340A-104 only

■ Block Diagram of Flash/Mask ROM Version

Figure 1.2-3 Block Diagram of Flash/Mask ROM Version



*1: Product without S-suffix
*2: Product with T-suffix
*3: CR oscillation circuit/clock supervisor supports MB90367/T(S), MB90F367/T(S) only

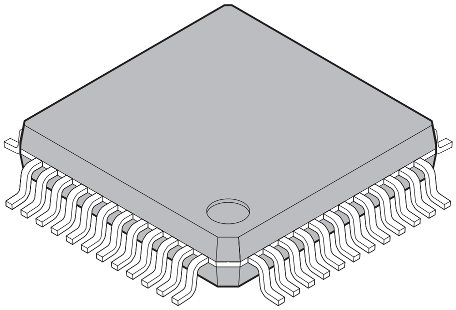
1.3 Package Dimensions

MB90360 series has a package.
Note that the dimensions show below are reference dimensions. For formal dimensions of each package, contact us.

■ Package Dimensions

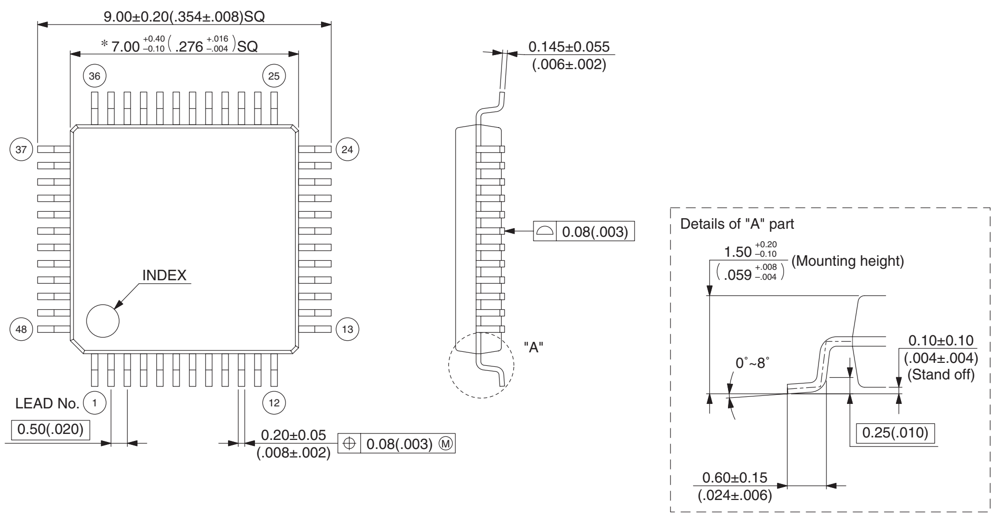
Figure 1.3-1 shows the package dimensions of LQFP-48 type.

Figure 1.3-1 Package Dimensions of LQFP-48 Type

<table>
<tr><td rowspan="7">48-pin plastic LQFP

(FPT-48P-M26)</td><td>Lead pitch</td><td>0.50 mm</td></tr>
<tr><td>Package width × package length</td><td>7 × 7 mm</td></tr>
<tr><td>Lead shape</td><td>Gullwing</td></tr>
<tr><td>Sealing method</td><td>Plastic mold</td></tr>
<tr><td>Mounting height</td><td>1.70 mm MAX</td></tr>
<tr><td>Weight</td><td>0.17 g</td></tr>
<tr><td>Code (Reference)</td><td>P-LFQFP48-7×7-0.50</td></tr>
</table>

48-pin plastic LQFP
(FPT-48P-M26)

Note 1) ∗ : These dimensions include resin protrusion.
Note 2) Pins width and pins thickness include plating thickness.
Note 3) Pins width do not include tie bar cutting remainder.



Ⓒ 2003 FUJITSU LIMITED F48040S-c-2-2

Dimensions in mm (inches).
Note: The values in parentheses are reference values.

1.4 Pin Assignment

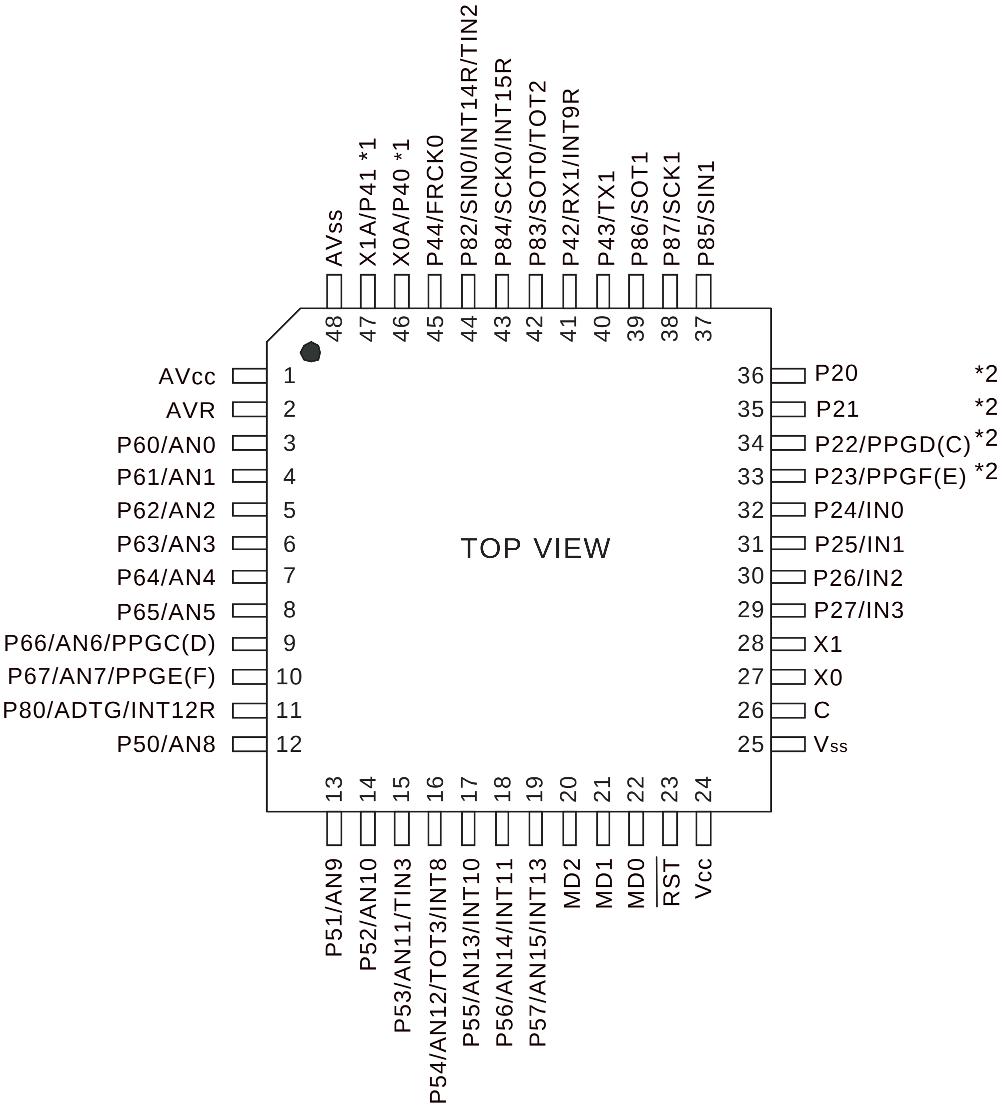
This section shows the pin assignments for the MB90360 series.

■ Pin assignment (LQFP-48)

Figure 1.4-1 shows the pin assignments of LQFP-48 type.

Figure 1.4-1 Pin Assignment (LQFP-48)

(TOP VIEW)



(FPT-48P-M26)

*1: MB90F362/T, MB90362/T, MB90F367/T, MB90367/T : X0A, X1A
MB90F362S/TS, MB90362S/TS, MB90F367S/TS, MB90367S/TS : P40, P41

*2: High current port

1.5 Pin Functions

Table 1.5-1 describes the pin functions of the MB90360 series.

■ Pin Functions

Table 1.5-1 Pin Description (1/3)

Pin number	Pin name	Circuit type	Functional description
1	AV_{CC}	I	V_{CC} power input pin for analog circuit
2	AVR	-	Power (Vref+) input pin for A/D converter. The power supply should not be input V_{CC} exceeding.
3 to 8	P60 to P65	H	General-purpose I/O port
	AN0 to AN5		Analog input pin for A/D converter.
9 to 10	P66, P67	H	General-purpose I/O port
	AN6, AN7		Analog input pin for A/D converter
	PPGC(D), PPGE(F)		Output pin for PPG
11	P80	F	General-purpose I/O port
	ADTG		Trigger input pin for A/D converter
	INT12R		External interrupt request input pin for INT12R.
12 to 14	P50 to P52	H	General-purpose I/O port (I/O circuit type of P50 is different from that of MB90V340A)
	AN8 to AN10		Analog input pin for A/D converter
15	P53	H	General-purpose I/O port
	AN11		Analog input pin for A/D converter
	TIN3		Event input pin for reload timer 3
16	P54	H	General-purpose I/O port
	AN12		Analog input pin for A/D converter
	TOT3		Output pin for reload timer 3
	INT8		External interrupt request input pin for INT8
17 to 19	P55 to P57	H	General-purpose I/O port
	AN13 to AN15		Analog input pin for A/D converter
	INT10, INT11, INT13		External interrupt request input pin for INT10, INT11 and INT13
20	MD2	D	Input pin for selecting operation mode
21, 22	MD1,MD0	C	Input pin for selecting operation mode
23	RST	E	Reset input
24	V_{CC}	-	Power input pin (3.5 V to 5.5 V)
25	V_{SS}	-	Power input pin (0 V)

Table 1.5-1 Pin Description (2/3)

Pin number	Pin name	Circuit type	Functional description
26	C	I	Capacity pin for stabilizing power supply. It should be connected to higher than or equal to 0.1 µF ceramic capacitor.
27	X0	A	Oscillation input pin
28	X1	A	Oscillation output pin
29 to 32	P24 to P27	G	General-purpose I/O port The register can be set to select whether to use pull-up register. This function is enabled in single-chip mode.
	IN0 to IN3		Event input pin for input capture 0 to 3.
33, 34	P22 to P23	J	General-purpose I/O port The pull-up resistor ON/OFF can be set by setting the register. This function becomes valid at shingle-chip mode. High current output port
	PPGF(E), PPGD(C)		Output pin for PPG
35, 36	P20, P21	J	General-purpose I/O port The pull-up resistor ON/OFF can be set by setting the register. This function becomes valid at shingle-chip mode. High current output port
37	P85	K	General-purpose I/O port
	SIN1		Serial data input pin for UART1
38	P87	F	General-purpose I/O port
	SCK1		Clock I/O pin for UART1
39	P86	F	General-purpose I/O port
	SOT1		Serial data output pin for UART1
40	P43	F	General-purpose I/O port
	TX1		TX output pin for CAN1 interface
41	P42	F	General-purpose I/O port
	RX1		RX input pin for CAN1 interface
	INT9R		External interrupt request input pin for INT9R (sub)
42	P83	F	General-purpose I/O port
	SOT0		Serial data output pin for UART0
	TOT2		Output pin for reload timer 2
43	P84	F	General-purpose I/O port
	SCK0		Clock I/O pin for UART0
	INT15R		External interrupt request input pin for INT15R
44	P82	K	General-purpose I/O port
	SIN0		Serial data input pin for UART0
	INT14R		External interrupt request input pin for INT14R
	TIN2		Event input pin for reload timer 2

Table 1.5-1 Pin Description (3/3)

Pin number	Pin name	Circuit type	Functional description
45	P44	F	General-purpose I/O port (I/O circuit type of P44 is different from that of MB90V340A.)
	FRCK0		Free-run timer 0 clock pin
46, 47	P40, P41	F	General-purpose I/O port (product with S-suffix and MB90V340A-101/103 only)
	X1A, X0A	B	Oscillation input pin for subclock (product without S-suffix and MB90V340A-102/104 only)
48	AV$_{SS}$	I	V$_{SS}$ power input pin for analog circuit

1.6 Input-Output Circuits

Table 1.6-1 lists the input-output circuits.

■ Input-output Circuits

Table 1.6-1 I/O Circuit Types (1/4)

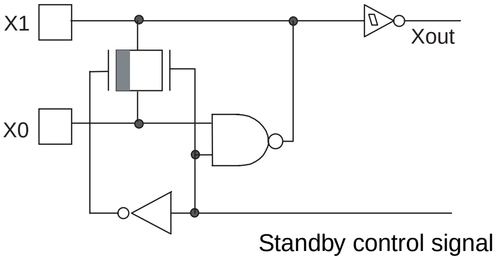
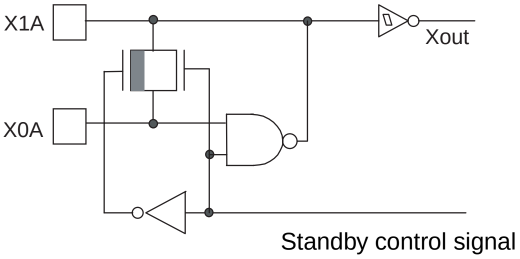
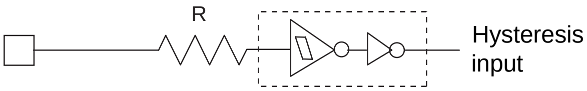
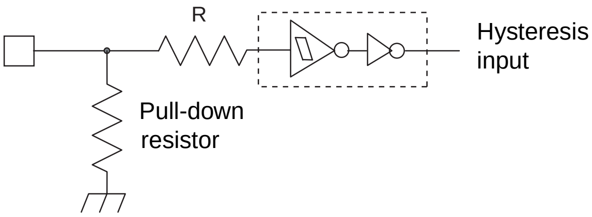
Type	Circuit	Remarks
A		Oscillation circuit High-speed oscillation feedback resistor = approx. 1 MΩ
B		Oscillation circuit Low-speed oscillation feedback resistor = approx. 10 MΩ
C		Mask ROM device : CMOS hysteresis input pin Flash device: CMOS input
D		Mask ROM device : CMOS hysteresis input pin Pull-down resistor value: approx. 50 kΩ Flash device: CMOS input pin No Pull-down

Table 1.6-1 I/O Circuit Types (2/4)

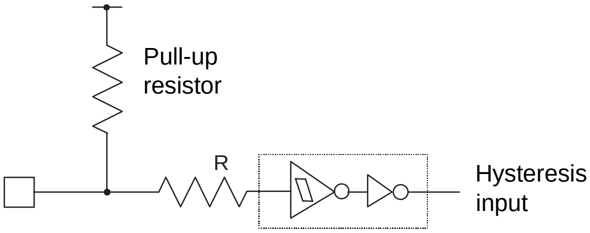
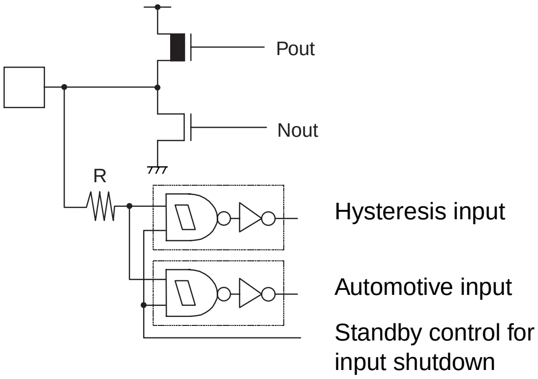
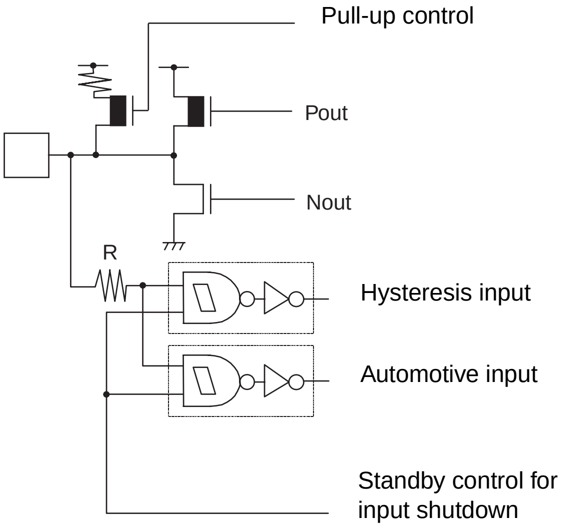
Type	Circuit	Remarks
E		CMOS hysteresis input pin Pull-up resister value: approx. 50 kΩ
F	 Pout Nout R Hysteresis input Automotive input Standby control for input shutdown	CMOS level output (I_{OL} = 4 mA, I_{OH} =-4 mA) CMOS hysteresis inputs (with the standby-time input shutdown function) Automotive input (with the standby-time input shutdown function)
G	 Pull-up control Pout Nout R Hysteresis input Automotive input Standby control for input shutdown	CMOS level output (I_{OL} = 4 mA, I_{OH} =-4 mA) CMOS hysteresis inputs (with the standby-time input shutdown function) Automotive input (with the standby-time input shutdown function) Programmable pull-up resistor: approx. 50 kΩ

Table 1.6-1 I/O Circuit Types (3/4)

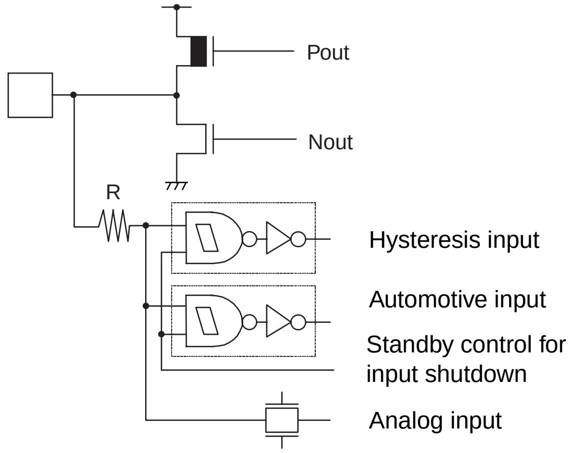
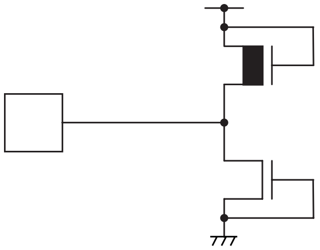
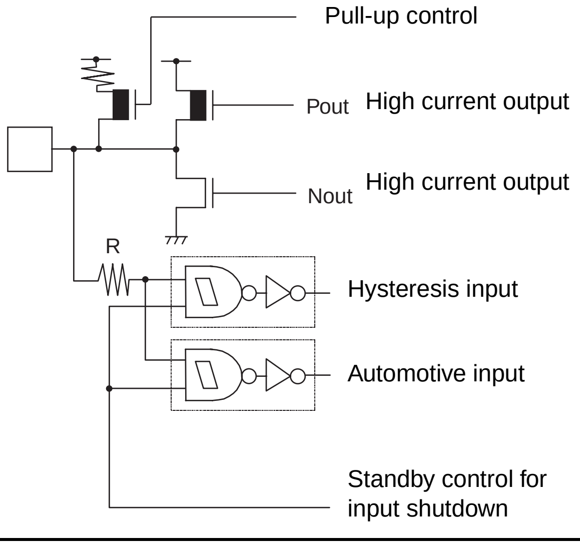
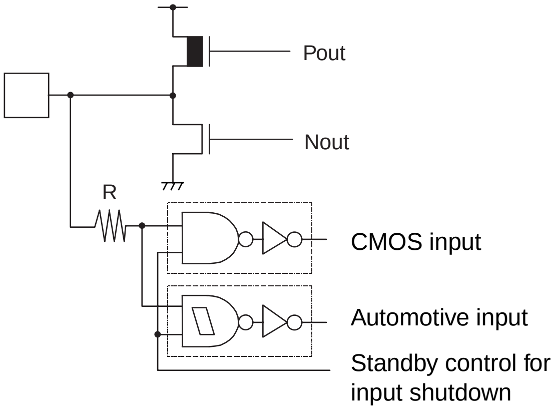
Type	Circuit	Remarks
H	 Pout Nout R Hysteresis input Automotive input Standby control for input shutdown Analog input	CMOS level output (I_{OL} = 4 mA, I_{OH} =-4 mA) CMOS hysteresis inputs (with the standby-time input shutdown function) Automotive input (with the standby-time input shutdown function) A/D analog input
I		Power supply input protection circuit
J	 Pull-up control Pout High current output Nout High current output R Hysteresis input Automotive input Standby control for input shutdown	CMOS level output (I_{OL} = 20 mA, I_{OH} =-14 mA) CMOS hysteresis inputs (with the standby-time input shutdown function) Automotive inputs (with the standby-time input shutdown function) Programmable pull-up resistor: approx. 50 kΩ

Table 1.6-1 I/O Circuit Types (4/4)

Type	Circuit	Remarks
K	 Pout Nout R CMOS input Automotive input Standby control for input shutdown	CMOS level output (I_{OL} = 4 mA, I_{OH} = -4 mA) CMOS hysteresis inputs (with the standby-time input shutdown function) Automotive hysteresis inputs (with the standby-time input shutdown function)

1.7 Handling Device

This section explains notes on handling the MB90360 series.

■ Handling the Device

● Preventing latch-up

CMOS IC chips may suffer latch-up under the following conditions:

- A voltage higher than V_{CC} or lower than V_{SS} is applied to an input or output pin.
- A voltage higher than the rated voltage is applied between V_{CC} and V_{SS}.
- The AV_{CC} power supply is applied before the V_{CC} voltage.

Latch-up may increase the power supply current drastically, causing thermal damage to the device.

When used, note that maximum rated voltage is not exceeded.

For the same reason, also be careful not to let the analog power-supply voltage ($AV_{CC,}$ AVR) exceed the digital power-supply voltage.

● Treatment of unused pins

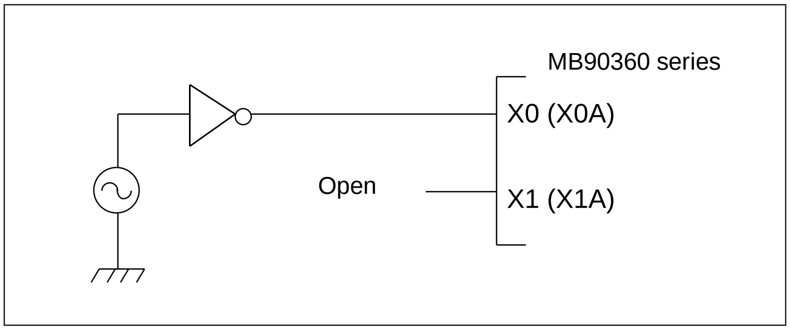
Leaving unused input pins open may result in misbehavior or latch up and possible permanent damage of the device. Therefore, they must be pulled up or pulled down through resistors. In this case those resistors should be more than 2 kΩ.

Unused bidirectional pins should be set to the output state and can be left open, or the input state with the above described connection.

● Using external clock

To use external clock, drive the X0 (X0A) pin and leave X1 (X1A) pin open.

Figure 1.7-1 Using External Clock



● Precautions for when not using a sub clock signal

If you do not connect pins X0A and X1A to an oscillator, use pull-down handling on the X0A pin, and leave the X1A pin open.

● Notes on during operation of PLL clock mode

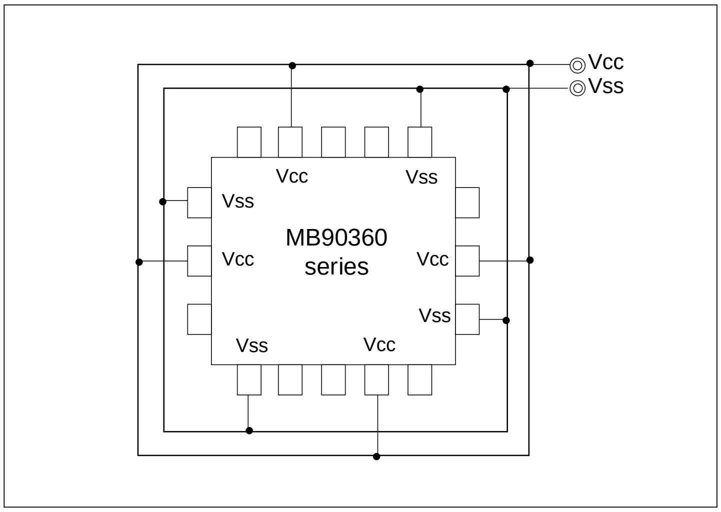
If the PLL clock mode is selected, the microcontroller attempts to be working with the free-running frequency of self-oscillating circuit in the PLL even when there is no external oscillator or external clock input is stopped. Performance of this operation, however, cannot be guaranteed.

● Power supply pins (V_{CC}/V_{SS})

- If there are multiple V_{CC} and V_{SS} pins, from the point of view of device design, pins to be of the same potential are connected the inside of the device to prevent such malfunctioning as latch up.

  To reduce unnecessary radiation, prevent malfunctioning of the strobe signal due to the rise of ground level, and to keep the recommended DC characteristics specified as the total output current, be sure to connect the V_{CC} and V_{SS} pins to the power supply and ground externally (see Figure 1.7-2).
- Connect V_{CC} and V_{SS} to the device from the power supply source with lowest possible impedance.
- It is recommended to connect a capacitor of about 0.1 µF as a bypass capacitor between V_{CC} and V_{SS} in the vicinity of V_{CC} and V_{SS} pins of the device

Figure 1.7-2 Power Supply Pins (V_{CC}/V_{SS})



● Pull-up/down resistors

The MB90360 Series does not support internal pull-up/down resistors (except Port2: programmable pull-up resistors). Use pull-up/down handling where needed.

● Crystal Oscillator Circuit

Noises around X0 or X1 pins may be possible causes of abnormal operations. Make sure to provide bypass capacitors via shortest distance from X0, X1 pins, crystal oscillator (or ceramic resonator) and ground lines, and make sure, to the utmost effort, that lines of oscillation circuit not cross the lines of other circuits.

It is highly recommended to provide a printed circuit board art work surrounding X0 and X1 pins with a ground area for stabilizing the operation.

● Turning-on Sequence of Power Supply to A/D Converter and Analog Inputs

Make sure to turn on the A/D converter power supply (AV_{CC}, AVR) and analog inputs (AN0 to AN15) after turning-on the digital power supply (V_{CC}).

Turn-off the digital power supply after turning off the A/D converter power supply and analog inputs. In this case, make sure that the voltage not exceed AVR or AV_{CC}.

● Connection of Unused Pins of A/D Converter

Connect unused pins of A/D converter as $AV_{CC} = V_{CC}$, $AV_{SS} = AVR = V_{SS}$.

● Notes on Energization

To prevent malfunction of the internal voltage regulator, supply voltage profile while turning on the power supply should be slower than 50 µs from 0.2 V to 2.7 V.

● Stabilization of power supply voltage

If the power supply voltage varies acutely even within the operation assurance range of the V_{CC} power supply voltage, a malfunction may occur. The V_{CC} power supply voltage must therefore be stabilized. As stabilization guidelines, stabilize the power supply voltage so that V_{CC} ripple fluctuations (peak to peak value) in the commercial frequencies (50 to 60 Hz) fall within 10% of the standard V_{CC} power supply voltage and the transient fluctuation rate becomes 0.1 V/ms or less in instantaneous fluctuation for power supply switching.

● Note on using CAN Function

The MB90360 series does not contain the clock modulation function. So, at using CAN, the DIRECT bit of the CAN direct mode register (CDMR) must be set "1". (See Table 1.7-1). If the DIRECT bit is not set correctly, the device does not operate normally.

Table 1.7-1 Setting of Clock Modulation and CAN Direct Mode

	Clock modulation setting (CMCR:PMOD bit)	CAN direct mode setting (setting of CDMR:DIRECT bit)
Required setting	0: Disable clock modulation (initial value)	1: Enable CAN direct mode
Setting disabled	1: Enable clock modulation	0: Disable CAN direct mode (initial value)

Note:

For details on the clock modulation, see "CHAPTER 6 CLOCK SUPERVISOR".
For details on the CAN direct mode, see "21.11 Setting Configuration of Multi-level Message Buffer".

● Flash security Function

The security bit is located in the area of the flash memory.
If protection code 01_H is written in the security bit, the flash memory is in the protected state by security.
Therefore please do not write 01_H in this address if you do not use the security function.
Please refer to following table for the address of the security bit.

	Flash memory size	Address of security bit
MB90F362/T(S), MB90F367/T(S)	Embedded 512Kbit flash memory	$FF0001_H$

● For the diversion of MB90340-series software assets

In programming of the MB90360 series, keep the following points in mind for the diversion of MB90340-series software assets in particular.

- Access to the registers which do not exist in the MB90360 series.

  As for the registers and bits which exist in MB90340 series but not in the MB90360 series, do not access them or ensure that the initial value is set. Setting any other value than the initial value may cause an abnormal operation in emulation using MB90V340.

- Setting of the external interrupt factor select register (EISSR).

  The MB90360 series is not equipped with the external interrupt request input, INT8R, INT9, INT10, INT11, INT12, INT13, INT14 and INT15.
  Enabling unequipped terminals causes a false operation. First set the EISSR and then set each of the registers when DTP/external interrupt is used.

CHAPTER 2

CPU

This chapter explains the CPU.

2.1 Outline of the CPU

The F^{2}MC-16LX CPU core is a 16-bit CPU designed for applications that require high-speed real-time processing, such as home-use or vehicle-mounted electronic appliances. The F^{2}MC-16LX instruction set is designed for controller applications, and is capable of high-speed, highly efficient control processing.

■ Outline of the CPU

In addition to 16-bit data, the F^{2}MC-16LX CPU core can process 32-bit data by using an internal 32-bit accumulator. (32-bit data can be processed with some instructions.) Up to 16M bytes of memory space (expandable) can be used, which can be accessed by either the linear pointer or bank method. The instruction system, based on the F^{2}MC-8 A-T architecture, has been reinforced by adding instructions compatible with high-level languages, expanding addressing modes, reinforcing multiplication and division instructions, and enhancing bit processing. The features of the F^{2}MC-16LX CPU are explained below.

● Minimum instruction execution time: 42 ns (at 4-MHz oscillation, 6 times clock multiplication)

● Maximum memory space: 16M bytes, accessed in linear or bank mode

● Instruction set optimized for controller applications

- Rich data types: Bit, byte, word, long word
- Extended addressing modes: 23 types
- High-precision operation (32-bit length) based on 32-bit accumulator

● Powerful interrupt functions

Eight priority levels (programmable)

● CPU-independent automatic transfer

Up to 16 channels of the extended intelligent I/O service

● Instruction set compatible with high-level language (C)/multitasking

System stack pointer/instruction set symmetry/barrel-shift instructions

● Improved execution speed: 4 bytes queue

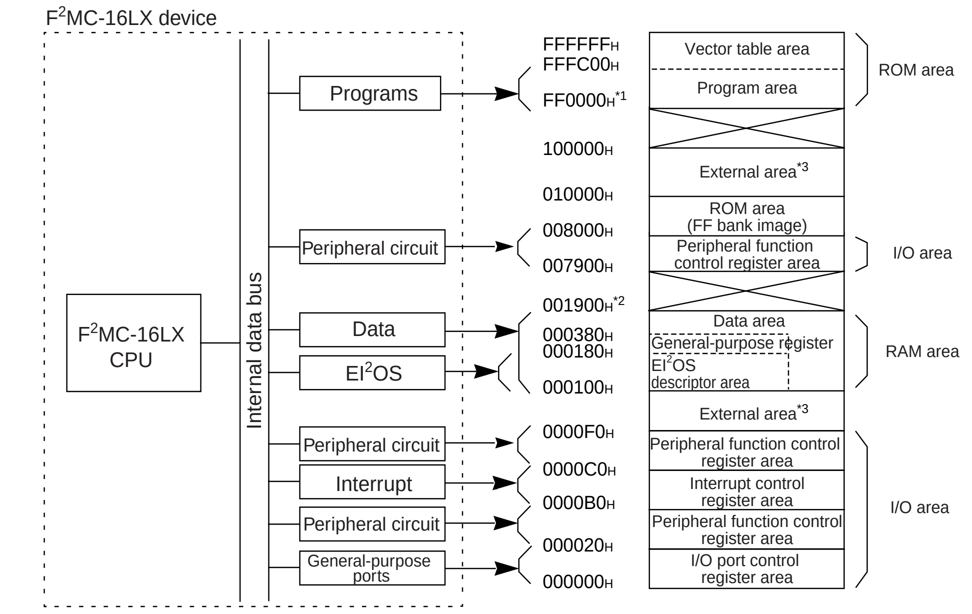
2.2 Memory Space

An F^{2}MC-16LX CPU has a 16M bytes memory space. All data program input and output managed by the F^{2}MC-16LX CPU are located in this 16M bytes memory space. The CPU accesses the resources by indicating their addresses using a 24-bit address bus.

■ Outline of CPU Memory Space

Figure 2.2-1 shows a sample relationship between the F^{2}MC-16LX system and memory map.

Figure 2.2-1 Sample Relationship between F^{2}MC-16LX System and Memory Map



*1: The size of the internal ROM differs for each model.
*2: The size of the internal RAM differs for each model.
*3: Access is not possible in single-chip mode.

■ ROM area

● Vector table area (address: FFFC00$_H$ to FFFFFF$_H$)

This area is used as a vector table for reset/interrupt and CALLV vector.

This area is allocated at the highest addresses of the ROM area. The start address of the corresponding processing routine is set as data in each vector table address.

● Program area (address: FF0000$_H$ to FFFBFF$_H$)

ROM is built in as an internal program area.

The size of internal ROM differs for each model.

■ RAM Area

● Data area (address: From 000100$_H$ to 000CFF$_H$ (for 3K bytes))

The static RAM is built in as an internal data area.

The size of internal RAM differs for each model.

● General-purpose register area (address: 000180$_H$ to 00037F$_H$)

Auxiliary registers used for 8-bit, 16-bit, and 32-bit arithmetic operations and transfer are allocated in this area.

Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.

When this area is used as a general-purpose register, general-purpose register addressing enables high-speed access with short instructions.

● Extended intelligent I/O service (EI^{2}OS) descriptor area (address: 000100$_H$ to 00017F$_H$)

This area retains the transfer modes, I/O addresses, transfer count, and buffer addresses.

Since this area is allocated to a part of the RAM area, it can be used as ordinary RAM.

■ I/O Area

● Interrupt control register area (address: 0000B0$_H$ to 0000BF$_H$)

The interrupt control registers (ICR00 to ICR15) correspond to all peripheral functions that have an interrupt function. These registers set interrupt levels and control the extended intelligent I/O service (EI^{2}OS).

● Peripheral function control register area (address: 000020$_H$ to 0000AF$_H$, 0000C0$_H$ to 0000EF$_H$, 007900$_H$ to 007FFF$_H$)

This register controls the built-in peripheral functions and inputs and outputs data.

● I/O port control register area (address: 000000$_H$ to 00001F$_H$)

This register controls I/O ports, and inputs and outputs data.

■ Address generation types

The F^2MC-16LX has the following 2 addressing modes:

● Linear addressing

An entire 24-bit address is specified by an instruction.

● Bank addressing.

The eight high-order bits of an address are specified by an appropriate bank register, and the remaining 16 low-order bits are specified by an instruction.

2.3 Memory Map

The memory map of the MB90360 Series is shown in Figure 2.3-1 .

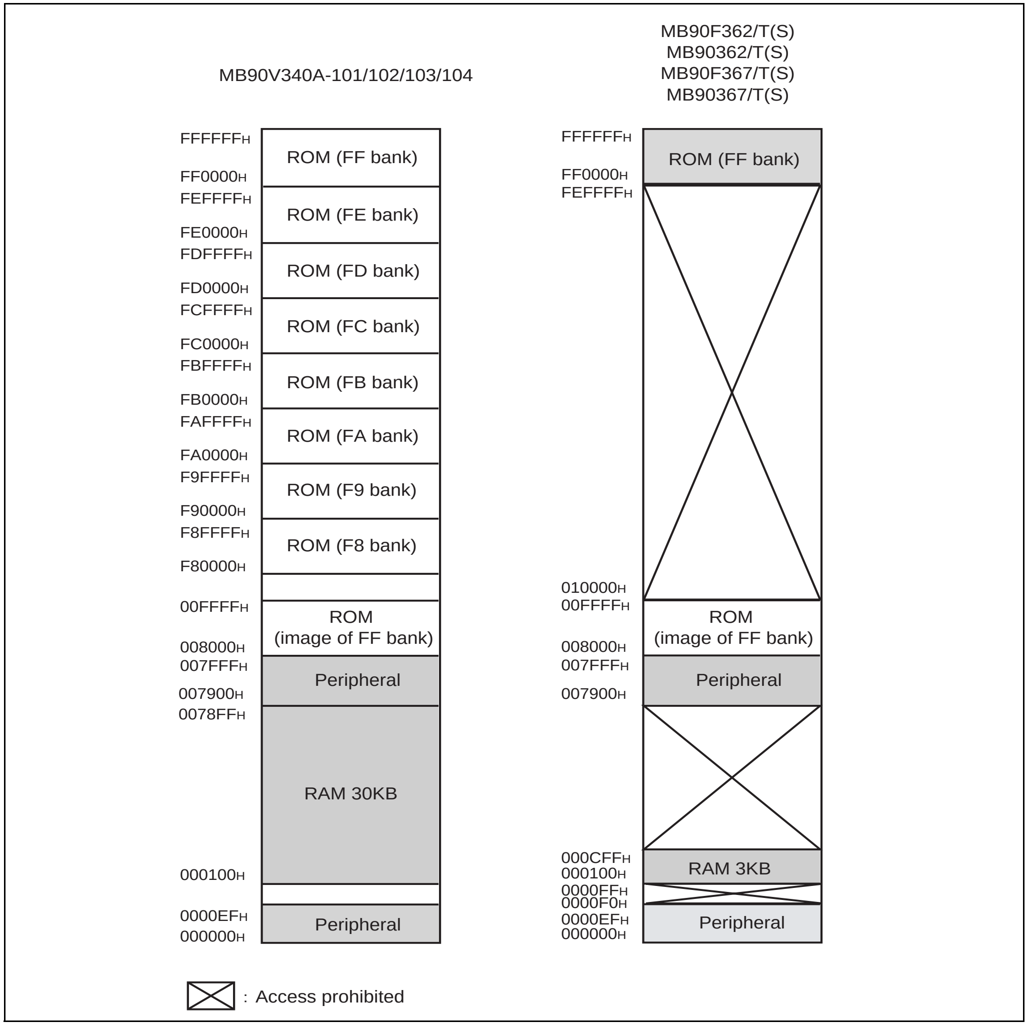
■ Memory Map

The ROM data in the high-order portion of FF-bank can be seen as an image in the higher 00-bank in order to support the small model C compiler. Since the low-order 16 bits are identical, this part of the ROM data can be referred without using the far specification in the pointer declaration.

For example, when 00C000$_H$ is accessed, the contents of ROM at FFC000$_H$ are read. However, since the ROM area in the FF bank exceeds 32K bytes, its entire image cannot be mirrored in the 00 bank.

The image between FF8000$_H$ and FFFFFF$_H$ is visible in bank 00, whereas the data between FF0000$_H$ and FF7FFF$_H$ is only visible in bank FF.

Figure 2.3-1 Memory Map



2.4 Linear Addressing

There are 2 types of linear addressing:

- **24-bit operand specification: Directly specifies a 24-bit address using operands.**
- **32-bit register indirect specification: Indirectly specifies the 24 low-order bits of a 32-bit general-purpose register value as the address.**

■ 24-bit Operand Specification

Figure 2.4-1 shows an example of 24-bit operand specification. Figure 2.4-2 shows an example of 32-bit register indirect specification.

Figure 2.4-1 Example of Linear Method (24-bit operand specification)

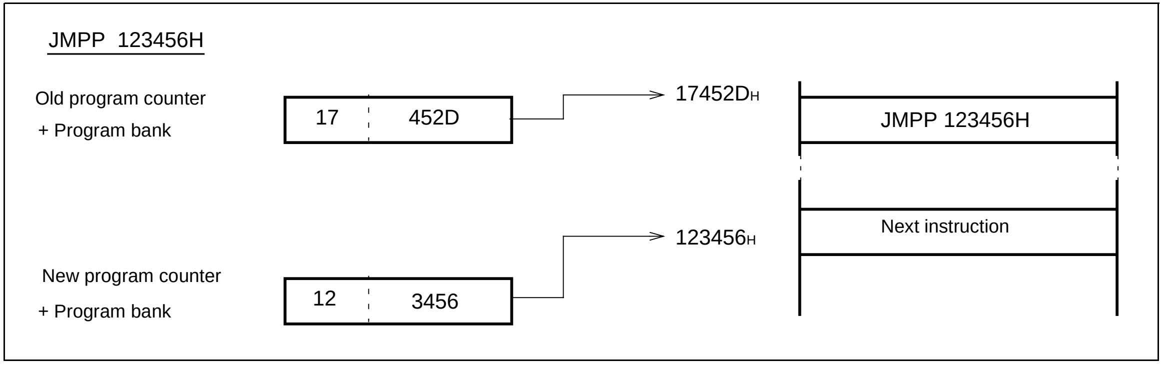
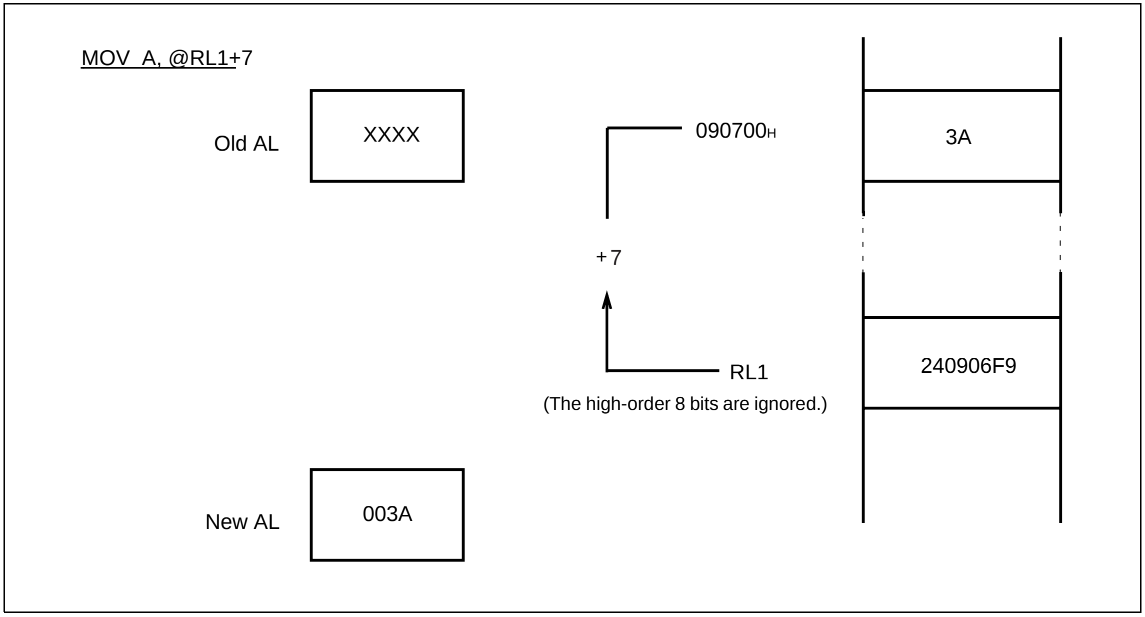


Figure 2.4-2 Example of Linear Method (32-bit register indirect specification)



2.5 Bank Addressing Types

In the bank method, the 16M bytes space is divided into 256 for 64K bytes banks. The following 5 bank registers are used to specify the banks corresponding to each space:

- **Program bank register (PCB)**
- **Data bank register (DTB)**
- **User stack bank register (USB)**
- **System stack bank register (SSB)**
- **Additional bank register (ADB)**

■ Bank Addressing Types

● Program bank register (PCB)

The 64K bytes bank specified by the PCB is called a program (PC) space. The PC space contains instruction codes, vector tables, and immediate value data, for example.

● Data bank register (DTB)

The 64K bytes bank specified by the DTB is called a data (DT) space. The DT space contains readable/writable data, and control/data registers for internal and external resources.

● User stack bank register (USB)/system stack bank register (SSB)

The 64K bytes bank specified by the USB or SSB is called a stack (SP) space. The SP space is accessed when a stack access occurs during a push/pop instruction or interrupt register saving. The S flag in the condition code register determines the stack space to be accessed.

● Additional bank register (ADB)

The 64K bytes bank specified by the ADB is called an additional (AD) space. The AD space, for example, contains data that cannot fit into the DT space.

Table 2.5-1 lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency. To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank space corresponding to the specified prefix code.

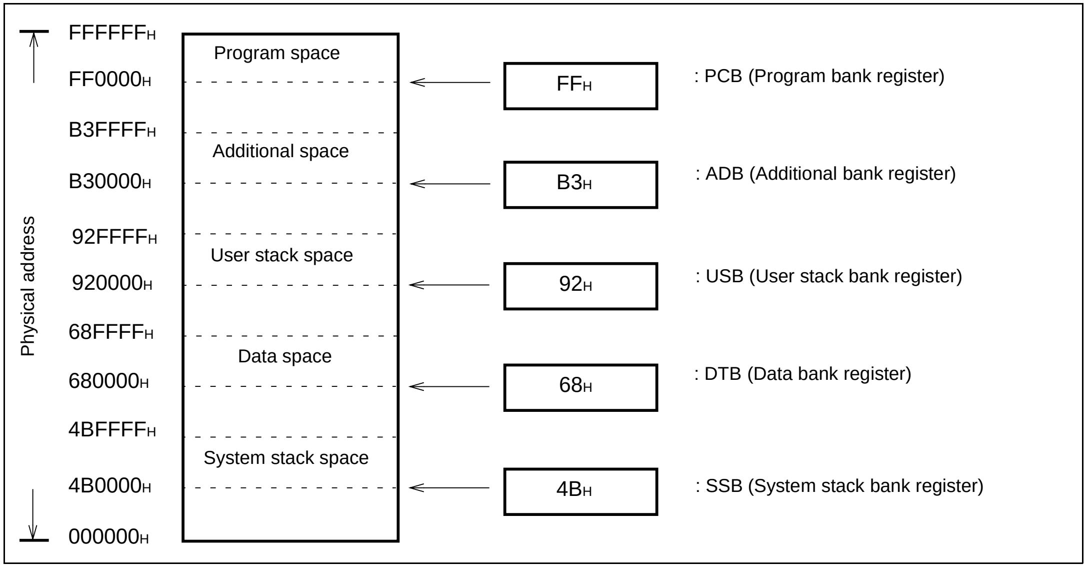
After reset, the DTB, USB, SSB, and ADB are initialized to 00_H. The PCB is initialized to a value specified by the reset vector. After reset, the DT, SP, and AD spaces are allocated in bank 00_H (000000_H to $00FFFF_H$), and the PC space is allocated in the bank specified by the reset vector.

Table 2.5-1 Default Space

Default space	Addressing mode
Program space	PC indirect, program access, branch
Data space	Addressing mode using @RW0, @RW1, @RW4, or @RW5, @A, addr16, and dir
Stack space	Addressing mode using PUSHW, POPW, @RW3, or @RW7
Additional space	Addressing mode using @RW2 or @RW6

Figure 2.5-1 is an example of a memory space divided into register banks.

Figure 2.5-1 Physical Addresses of Each Space



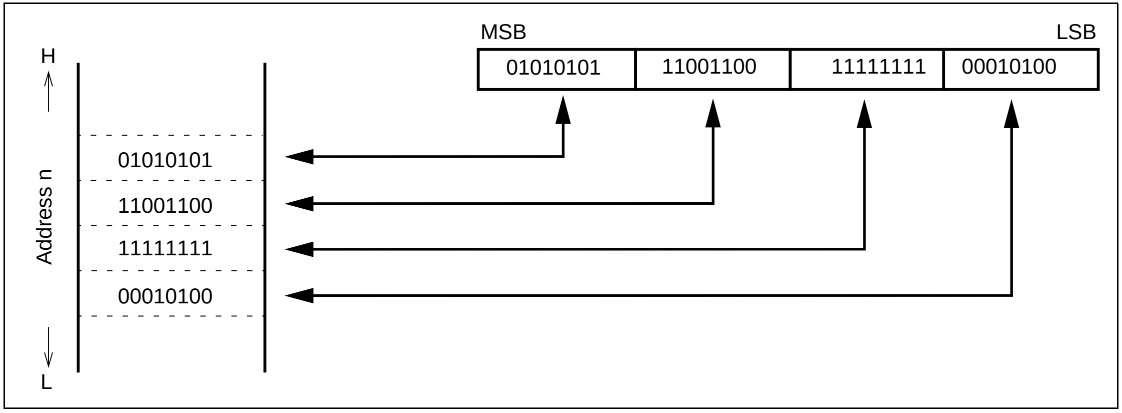
2.6 Multi-byte Data in Memory Space

Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the low-order 16 bits are transferred before the high-order 16 bits.
If a reset signal is inputted immediately after the low-order bits are written, the high-order bits might not be written.

■ Multi-byte Data Allocation in Memory Space

Figure 2.6-1 is a diagram of multi-byte data configuration in memory. The low-order eight bits of a data item are stored at address n, then address n+1, address n+2, address n+3, etc.

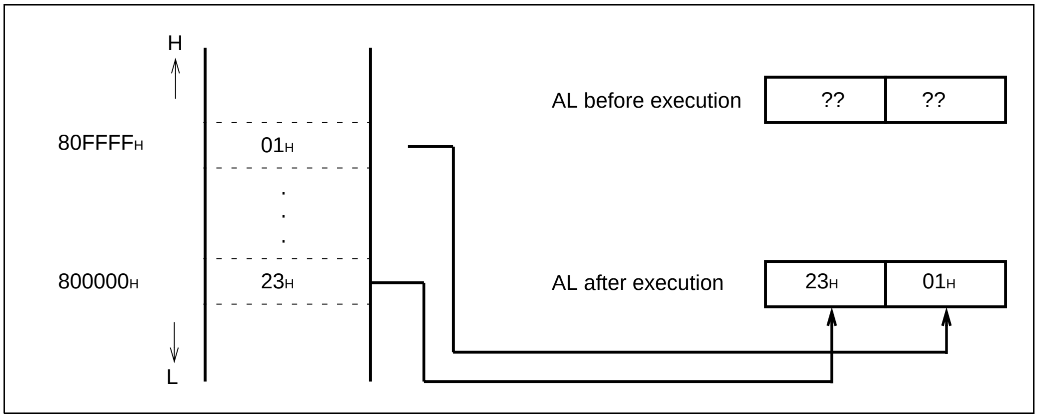
Figure 2.6-1 Sample Allocation of Multi-byte Data in Memory



■ Accessing Multi-byte Data

Fundamentally, accesses are made within a bank. For an instruction accessing a multi-byte data item, address FFFF$_H$ is followed by address 0000$_H$ of the same bank. Figure 2.6-2 is an example of an instruction accessing multi-byte data.

Figure 2.6-2 Execution of MOVW A, 080FFFF$_H$



2.7 Registers

The F^{2}MC-16LX registers are largely classified into two types: special registers in the CPU and general-purpose registers in memory. The special registers are dedicated internal hardware of the CPU, and they have specific use defined by the CPU architecture. The general-purpose registers share the CPU address space with RAM. The general-purpose registers are the same as the special registers in that they can be accessed without using an address. The applications of the general-purpose registers can be specified by the user however, as is ordinary memory space.

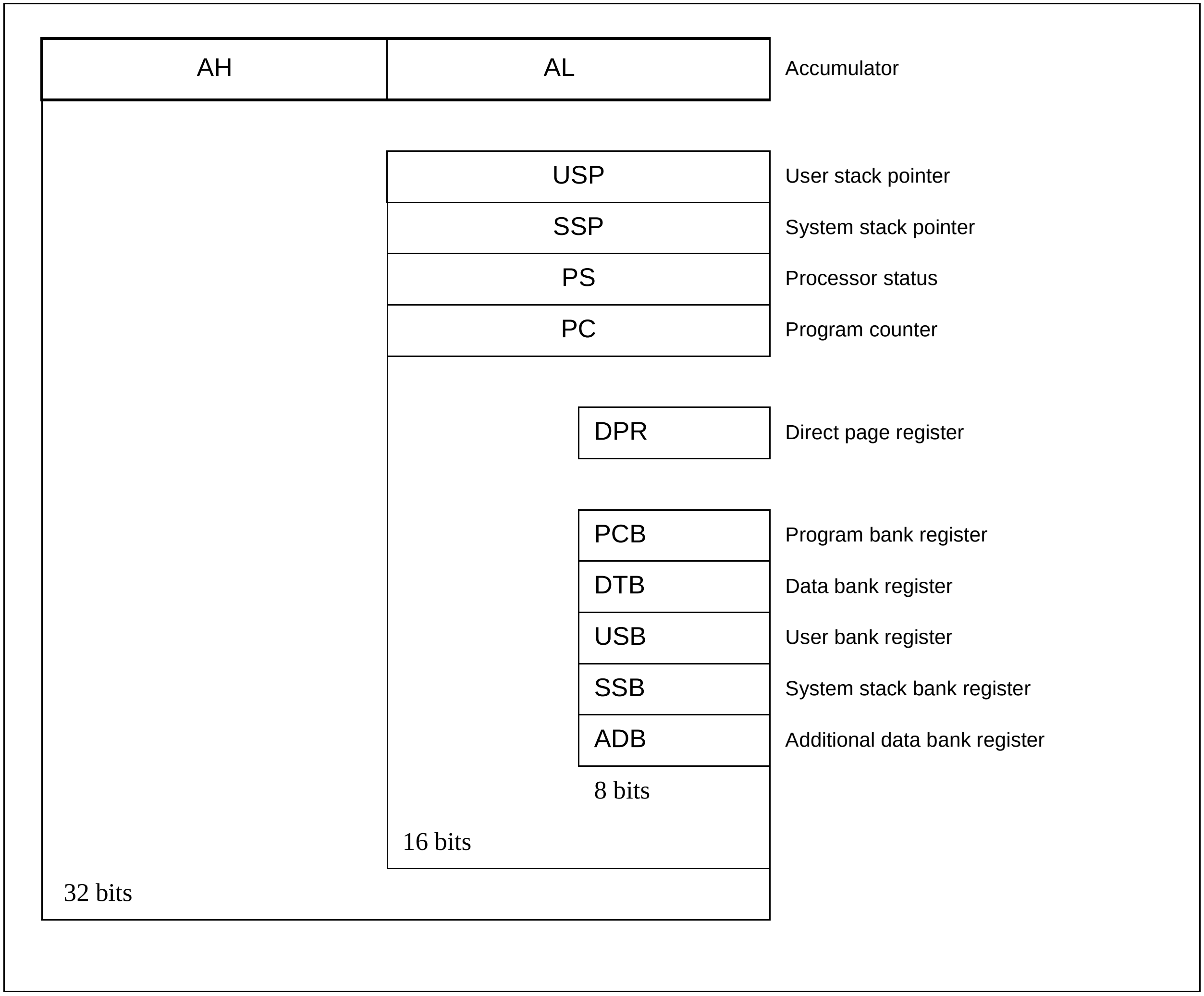
■ Special Registers

The F^{2}MC-16LX CPU core has the following special registers:

- Accumulator (A=AH:AL) : Two 16-bit accumulators (Can be used as a single 32-bit accumulator.)
- User stack pointer (USP) : 16-bit pointer indicating the user stack area
- System stack pointer (SSP) : 16-bit pointer indicating the system stack area
- Processor status (PS) : 16-bit register indicating the system status
- Program counter (PC) : 16-bit register holding the address of the program
- Program bank register (PCB) : 8-bit register indicating the PC space
- Data bank register (DTB) : 8-bit register indicating the DT space
- User stack bank register (USB) : 8-bit register indicating the user stack space
- System stack bank register (SSB): 8-bit register indicating the system stack space
- Additional bank register (ADB) : 8-bit register indicating the AD space
- Direct page register (DPR) : 8-bit register indicating a direct page

Figure 2.7-1 is a diagram of the special registers.

Figure 2.7-1 Special Registers

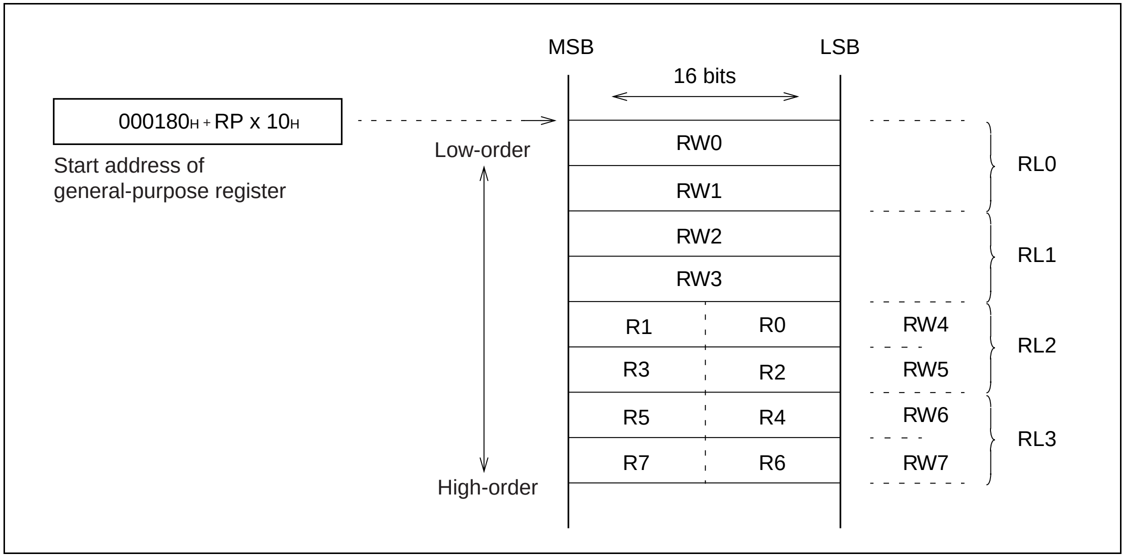


■ General-purpose registers

The F^{2}MC-16LX general-purpose registers are located from addresses 000180$_H$ to 00037F$_H$ (maximum configuration) of main storage. The register bank pointer (RP) indicates which of the above addresses are currently being used as a register bank. Each bank has the following three types of registers. These registers are mutually dependent as described in Figure 2.7-2 .

- R0 to R7: 8-bit general-purpose register
- RW0 to RW7: 16-bit general-purpose register
- RL0 to RL3: 32-bit general-purpose register

Figure 2.7-2 General-purpose Registers



The relationship between the high-order and low-order bytes of a byte or word register is expressed as follows:

RW $_{(i+4)}$ = R $_{(i \times 2+1)}$ $\times$ 256+R $_{(i \times 2)}$ [i=0 to 3]

The relationship between the high-order and low-order bytes of RLi and RW can be expressed as follows:

RL $_{(i)}$ = RW $_{(i \times 2+1)}$ $\times$ 65536+RW $_{(i \times 2)}$ [i=0 to 3]

2.7.1 Accumulator (A)

The accumulator (A) register consists of 2 16-bit arithmetic operation registers (AH and AL), and is used as a temporary storage for operation results and transfer data.

■ Accumulator (A)

During 32-bit data processing, AH and AL are used together. Only AL is used for word processing in 16-bit data processing mode or for byte processing in 8-bit data processing mode (see Figure 2.7-3 and Figure 2.7-4). The data stored in the A register can be operated upon with the data in memory or registers (Ri, Rwi, or RLi). In the same manner as with the F^{2}MC-8L, when a word or shorter data item is transferred to AL, the previous data item in AL is automatically sent to AH (data preservation function). The data preservation function and operation between AL and AH help improve processing efficiency.

When a byte or shorter data item is transferred to AL, the data is sign-extended or zero-extended and stored as a 16-bit data item in AL. The data in AL can be handled either as word or byte long.

When a byte-processing arithmetic operation instruction is executed on AL, the high-order eight bits of AL before operation are ignored. The high-order eight bits of the operation result all become zeroes.

The A register is not initialized by a reset. The A register holds an undefined value immediately after a reset.

Figure 2.7-3 32-bit Data Transfer

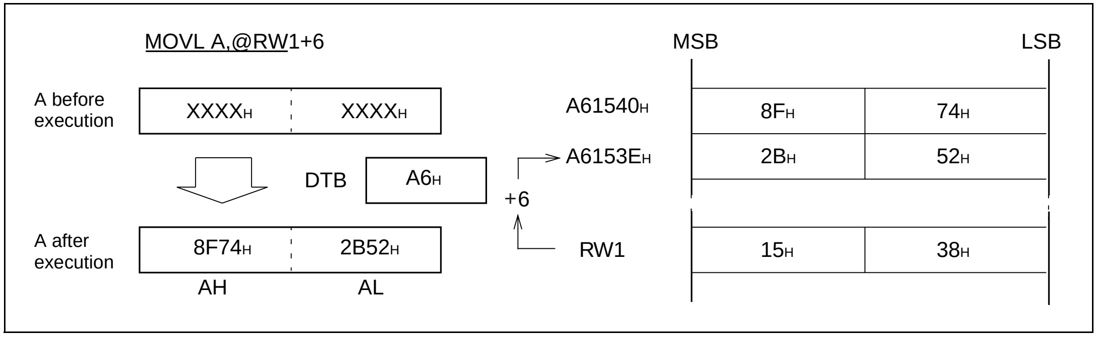
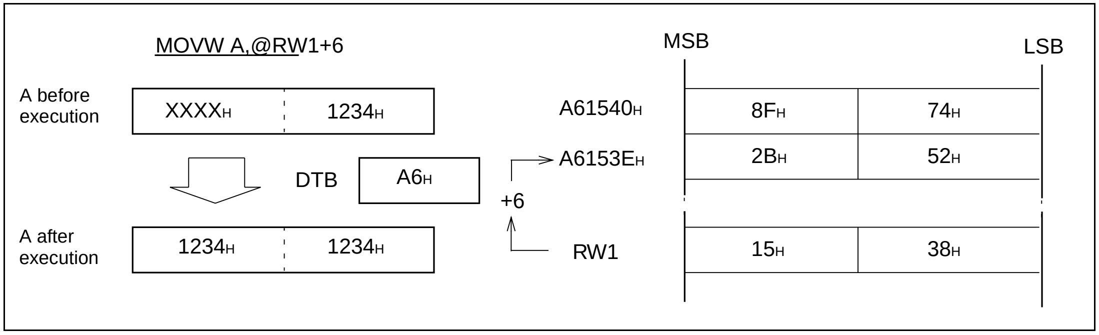


Figure 2.7-4 AL-AH Transfer



2.7.2 User Stack Pointer (USP) and System Stack Pointer (SSP)

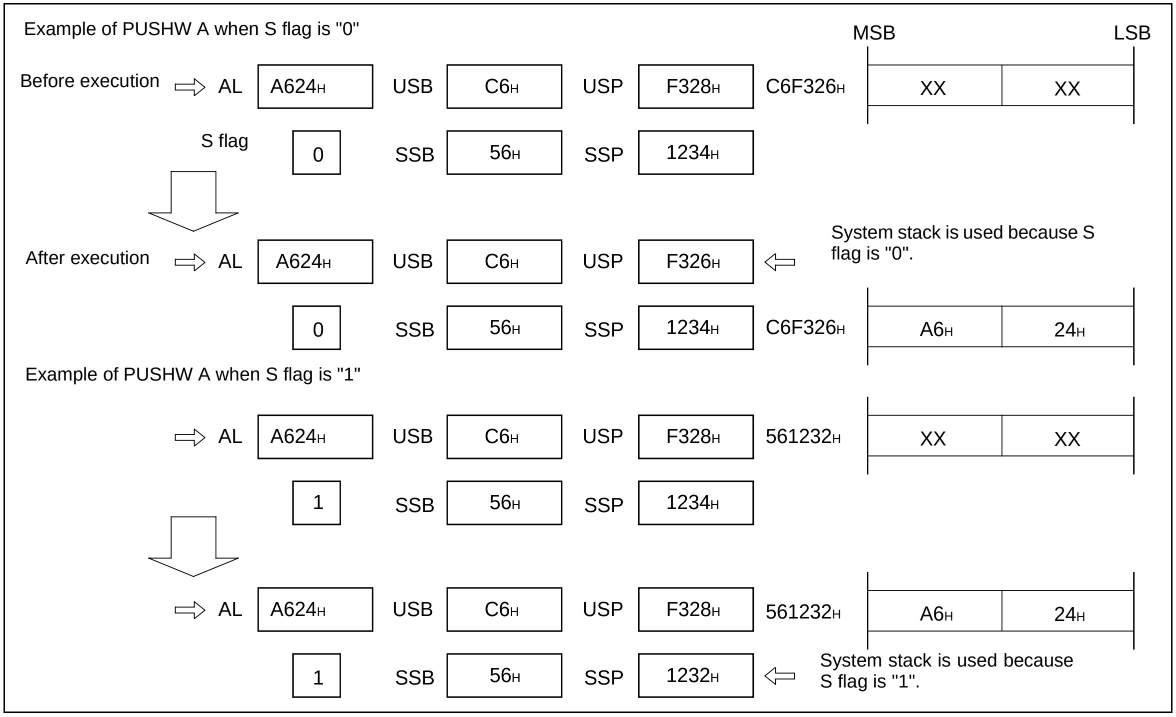
USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data when a push/pop instruction or subroutine is executed.

■ User Stack Pointer (USP) and System Stack Pointer (SSP)

The USP and SSP registers are used by stack instructions. The USP register is enabled when the S flag in the processor status register is “0”, and the SSP register is enabled when the S flag is “1” (see Figure 2.7-5). Since the S flag is set when an interrupt is accepted, register values are always saved in the memory area indicated by SSP during interrupt processing. SSP is used for stack processing in an interrupt routine, while USP is used for stack processing outside an interrupt routine. If the stack space is not divided, use only the SSP.

During stack processing, the high-order eight bits of an address are indicated by SSB (for SSP) or USB (for USP). USP and SSP are not initialized by a reset. Instead, they hold undefined values.

Figure 2.7-5 Stack Manipulation Instruction and Stack Pointer



Note:

Specify an even-numbered address in the stack pointer whenever possible.

2.7.3 Processor Status (PS)

The PS register consists of the bits controlling the CPU operation and the bits indicating the CPU status.

■ Processor Status (PS)

As shown in Figure 2.7-6 , the high-order byte of the PS register consists of a register bank pointer (RP) and an interrupt level mask register (ILM). The ILM indicates the start address of a register bank. The low-order byte of the PS register is a condition code register (CCR), containing the flags to be set or reset depending on the results of instruction execution or interrupt occurrences.

Figure 2.7-6 Processor Status (PS) Structure

	15 13	12 8	7 0
PS	ILM	RP	CCR

■ Condition Code Register (CCR)

Figure 2.7-7 is a diagram of condition code register (CCR) configuration.

Figure 2.7-7 Condition Code Register (CCR) Configuration

	7	6	5	4	3	2	1	0	
	-	I	S	T	N	Z	V	C	: CCR
Initial value	-	0	1	*	*	*	*	*	* : Undefined

● I: Interrupt enable flag:

Interrupt requests other than software interrupts are enabled when the I flag is 1 and are masked when the I flag is 0. The I flag is cleared by a reset.

● S: Stack flag:

When the S flag is 0, USP is enabled as the stack manipulation pointer.

When the S flag is 1, SSP is enabled as the stack manipulation pointer.

The S flag is set by an interrupt reception or a reset.

- T: Sticky bit flag:

1 is set in the T flag when there is at least one "1" in the data shifted out from the carry after execution of a logical right/arithmetic right shift instruction. Otherwise, 0 is set in the T flag. In addition, "0" is set in the T flag when the shift amount is zero.

- N: Negative flag:

The N flag is set when the MSB of the operation result is "1", and is otherwise cleared.

- Z: Zero flag:

The Z flag is set when the operation result is all zeroes, and is otherwise cleared.

- V: Overflow flag:

The V flag is set when an overflow of a signed value occurs as a result of operation execution and is otherwise cleared.

- C: Carry flag:

The C flag is set when a carry-up or carry-down from the MSB occurs as a result of operation execution, and is otherwise cleared.

■ Register Bank Pointer (RP)

The RP register indicates the relationship between the general-purpose registers of the F^2MC-16LX and the internal RAM addresses. Specifically, the RP register indicates the first memory address of the currently used register bank in the following conversion expression: [00180_H + (RP)*10_H] (see Figure 2.7-8). The RP register consists of five bits, and can take a value between 00_H and $1F_H$. Register banks can be allocated at addresses from 000180_H to 00037_H in memory.

Even within that range, however, the register banks cannot be used as general-purpose registers if the banks are not in internal RAM. The RP register is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the RP register; however, only the low-order five bits of that data are used.

Figure 2.7-8 Register Bank Pointer (RP)

	B4	B3	B2	B1	B0	: RP
Initial value	0	0	0	0	0	

■ Interrupt level mask register (ILM)

The ILM register consists of three bits, indicating the CPU interrupt masking level. An interrupt request is accepted only when the level of the interrupt is higher than that indicated by these three bits. Level 0 is the highest priority interrupt, and level 7 is the lowest priority interrupt (see Table 2.7-1). Therefore, for an interrupt to be accepted, its level value must be smaller than the current ILM value. When an interrupt is accepted, the level value of that interrupt is set in ILM. Thus, an interrupt of the same or lower level cannot be accepted subsequently. ILM is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the ILM register, but only the low-order three bits of that data are used.

Figure 2.7-9 Interrupt Level Mask Register (ILM)

	ILM2	ILM1	ILM0	: ILM
Initial value	0	0	0	

Table 2.7-1 Levels Indicated by the Interrupt Level Mask (ILM) Register

ILM2	ILM1	ILM0	Level value	Acceptable interrupt level
0	0	0	0	Interrupt disabled
0	0	1	1	0 only
0	1	0	2	Level value smaller than 1
0	1	1	3	Level value smaller than 2
1	0	0	4	Level value smaller than 3
1	0	1	5	Level value smaller than 4
1	1	0	6	Level value smaller than 5
1	1	1	7	Level value smaller than 6

2.7.4 Program Counter (PC)

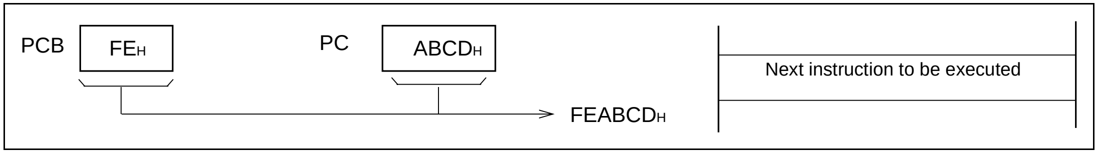
The PC register is a 16-bit counter that indicates the low-order 16 bits of the memory address of an instruction code to be executed by the CPU. The high-order eight bits of the address are indicated by the PCB. The PC register is updated by a conditional branch instruction, subroutine call instruction, interrupt, or reset.

The PC register can also be used as a base pointer for operand access.

■ Program Counter (PC)

Figure 2.7-10 shows the program counter.

Figure 2.7-10 Program Counter



2.8 Register Bank

A register bank consists of eight words. The register bank can be used as the following general-purpose registers for arithmetic operations: byte registers R0 to R7, word registers RW0 to RW7, and long word registers RL0 to RL3. In addition, the register bank can be used as instruction pointers.
RL0 to RL3 are used as the linear pointer that directly accesses entire space.

■ Register Bank

Table 2.8-1 lists the functions of the registers. Table 2.8-2 indicates the relationship between the registers.

In the same manner as for an ordinary RAM area, the register bank values are not initialized by a reset. The status before a reset is maintained. When the power is turned on, however, the register bank will have an undefined value.

Table 2.8-1 Register Functions

R0 to R7	Used as operands of instructions. Note: R0 is used as a counter for barrel shift and normalization instructions.
RW0 to RW7	Used as pointers. Used as operands of instructions. Note: RW0 is used as a counter for string instructions.
RL0 to RL3	Used as long pointers. Used as operands of instructions.

Table 2.8-2 Relationship between Registers

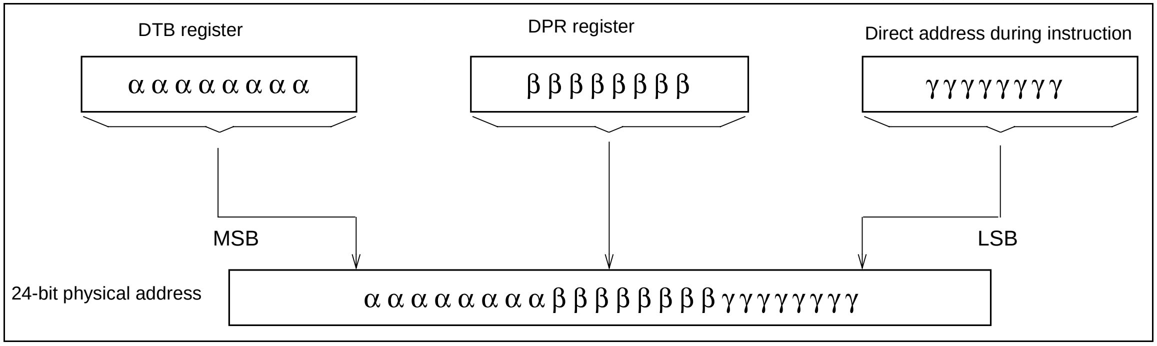
	RW0	RL0
	RW1	
	RW2	RL1
	RW3	
R0	RW4	RL2
R1		
R2	RW5	
R3		
R4	RW6	RL3
R5		
R6	RW7	
R7		

● Direct page register (DPR) <Initial value: 01_H>

DPR specifies addr8 to addr15 of the instruction operands in direct addressing mode as shown in Figure

2.8-1 . DPR is eight bits long, and is initialized to 01_H by a reset. DPR can be read or written to by an instruction.

Figure 2.8-1 Generating a Physical Address in Direct Addressing Mode



- Program counter bank register (PCB) <Initial value: Value in reset vector>

- Data bank register (DTB) <Initial value: 00_H>

- User stack bank register (USB) <Initial value: 00_H>

- System stack bank register (SSB) <Initial value: 00_H>

- Additional data bank register (ADB) <Initial value: 00_H>

Each bank register indicates the memory bank where the PC, DT, SP (user), SP (system), or AD space is allocated. All bank registers are one byte long. PCB is initialized to 00_H by a reset. Bank registers other than PCB can be read or written to. PCB can be read but cannot be written to.

PCB is updated when the JMPP, CALLP, RETP, RETIQ, or RETF instruction branching to the entire 16M bytes space is executed or when an interrupt occurs. For operation of each register, see "2.2 Memory Space".

2.9 Prefix Codes

Placing a prefix code before an instruction partially changes the operation of the instruction. Three types of prefix codes can be used: bank select prefix, common register bank prefix, and flag change disable prefix.

■ Bank Select Prefix

The memory space used for accessing data is determined for each addressing mode.

When a bank select prefix is placed before an instruction, the memory space used for accessing data by that instruction can be selected regardless of the addressing mode.

Table 2.9-1 lists the bank select prefixes and the corresponding memory spaces.

Table 2.9-1 Bank Select Prefix

Bank select prefix	Selected space
PCB	PC space
DTB	Data space
ADB	AD space
SPB	Either the SSP or USP space is used according to the stack flag value.

Use the following instructions with care:

- String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)

The bank register specified by an operand is used regardless of the prefix.

- Stack manipulation instructions (PUSHW, POPW)

SSB or USB is used according to the S flag regardless of the prefix.

- I/O access instructions

MOVA,io/MOVio,A/MOVXA,io/MOVWA,io/MOVWio,A/MOVio,#imm8
MOVWio,#imm16/MOVBA,io:bp/MOVBio:bp,A/SETBio:bp/CLRBio:bp
BBCio:bp,rel/BBSio:bp,rel/WBTC,WBTS
The IO space of the bank is used regardless of the prefix.

- Flag change instructions (AND CCR,#imm8, OR CCR,#imm8)

The instruction is executed normally, but the prefix affects the next instruction.

- POPW PS

SSB or USB is used according to the S flag regardless of the prefix. The prefix affects the next instruction.

- MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

- RETI

SSB is used regardless of the prefix.

■ Common Register Bank Prefix (CMR)

To simplify data exchange between multiple tasks, the same register bank must be accessed relatively easily regardless of the RP value. When CMR is placed before an instruction that accesses a register bank, the register accessed by that instruction can be changed to the common bank (the register bank selected when RP=0) at addresses from 000180_H to $00018F_H$ regardless of the current RP value. Use the following instructions with care:

- String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)

If an interrupt request occurs during execution of a string instruction with a prefix code, the prefix code becomes invalid when the string instruction is resumed after the interrupt is processed. Thus, the string instruction is executed falsely after the interrupt is processed. Do not prefix any of the above string instructions with CMR.

- Flag change instructions (AND CCR,#imm8, OR CCR,#imm8, POPW PS)

The instruction is executed normally, but the prefix affects the next instruction.

- MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

■ Flag Change Disable Prefix (NCC)

To disable flag changes, use the flag change disable prefix code (NCC). Placing NCC before an instruction that suppresses unnecessary flag change disables flag changes associated with that instruction. Use the following instructions with care:

- String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)

If an interrupt request occurs during execution of a string instruction with a prefix code, the prefix code becomes invalid when the string instruction is resumed after the interrupt is processed. Thus, the string instruction is executed incorrectly after the interrupt is processed. Do not prefix any of the above string instructions with NCC.

- Flag change instructions (AND CCR,#imm8, OR CCR,#imm8, POPW PS)

The instruction is executed normally, but the prefix affects the next instruction.

- Interrupt instructions (INT #vct8, INT9, INT addr16, INTP addr24, RETI)

CCR changes according to the instruction specifications regardless of the prefix.

- JCTX @A

CCR changes according to the instruction specifications regardless of the prefix.

● MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

2.10 Interrupt Disable Instructions

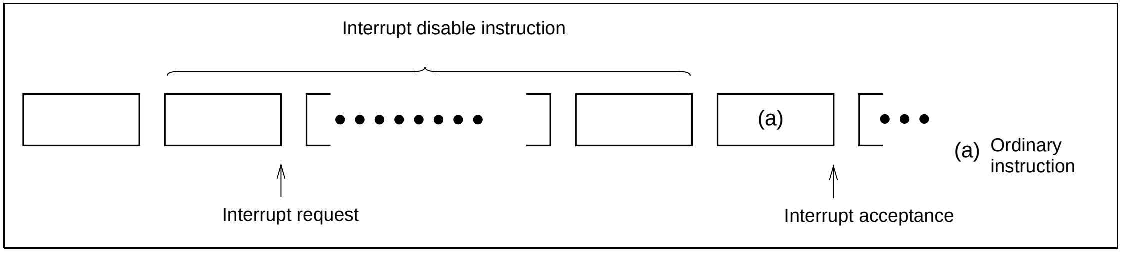
Interrupt requests are not sampled for the following ten instructions:

- MOV ILM,#imm8	**- PCB**	**- SPB**	**- OR CCR,#imm8**	**- NCC**
- AND CCR,#imm8	**- ADB**	**- CMR**	**- POPW PS**	**- DTB**

■ Interrupt Disable Instructions

If a valid hardware interrupt request occurs during execution of any of the above instructions, the interrupt can be processed only when an instruction other than the above is executed. For details, see Figure 2.10-1 .

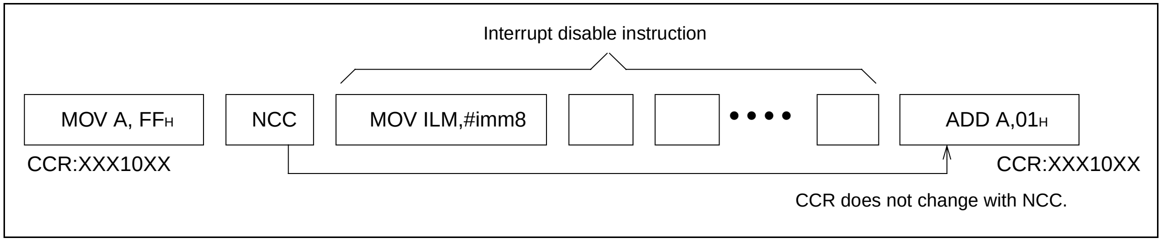
Figure 2.10-1 Interrupt Disable Instruction



■ Restrictions on Interrupt Disable Instructions and Prefix Instructions

When a prefix code is placed before an interrupt disable instruction, the prefix code affects the first instruction after the code other than the interrupt disable instruction. For details, see Figure 2.10-2 .

Figure 2.10-2 Interrupt Disable Instructions and Prefix Codes

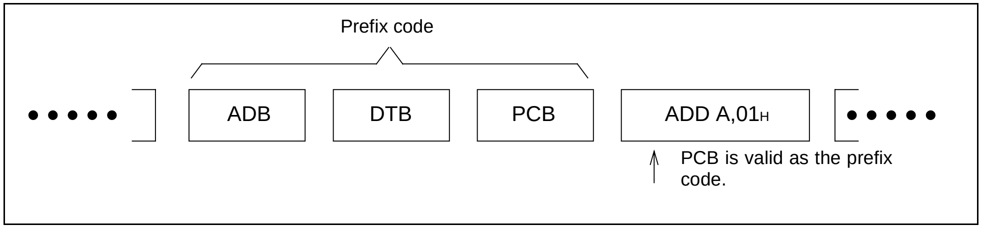


■ Consecutive prefix codes

When competitive prefix codes are placed consecutively, the latter becomes valid.

In the figure below, competitive prefix codes are PCB, ADB, DTB, and SPB.
For details, see Figure 2.10-3 .

Figure 2.10-3 Consecutive Prefix Codes



2.11 Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions

Set "00$_H$" in the bank register before using the "DIV A, Ri" and "DIVW A, RWi" instructions.

■ Precautions for Use of "DIV A, Ri" and "DIVW A, RWi" Instructions

Table 2.11-1 Precautions for Use of "DIVA,Ri" and "DIVWA,RWi" Instructions (i=0 to 7)

Instruction	Bank register name affected by the execution of the instructions listed on the left	Address that stores the remainder
DIVA,R0	DTB	(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+8$_H$ Lower 16 bits)
DIVA,R1		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+9$_H$ Lower 16 bits)
DIVA,R4		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+C$_H$ Lower 16 bits)
DIVA,R5		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+D$_H$ Lower 16 bits)
DIVWA,RW0		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+0$_H$ Lower 16 bits)
DIVWA,RW1		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+2$_H$ Lower 16 bits)
DIVWA,RW4		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+8$_H$ Lower 16 bits)
DIVWA,RW5		(DTB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+A$_H$ Lower 16 bits)
DIVA,R2	ADB	(ADB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+A$_H$ Lower 16 bits)
DIVA,R6		(ADB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+E$_H$ Lower 16 bits)
DIVWA,RW2		(ADB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+4$_H$ Lower 16 bits)
DIVWA,RW6		(ADB: Upper 8 bits)+(0180$_H$+RP × 10$_H$+E$_H$ Lower 16 bits)
DIVA,R3	USB, SSB*1	(USB*2: Upper 8 bits)+(0180$_H$+RP × 10$_H$+B$_H$ Lower 16 bits)
DIVA,R7		(USB*2: Upper 8 bits)+(0180$_H$+RP × 10$_H$+F$_H$ Lower 16 bits)
DIVWA,RW3		(USB*2: Upper 8 bits)+(0180$_H$+RP × 10$_H$+6$_H$ Lower 16 bits)
DIVWA,RW7		(USB*2: Upper 8 bits)+(0180$_H$+RP × 10$_H$+E$_H$ Lower 16 bits)

*1: Depends on the S bit of the CCR register
*2: In the event that S bit of the CCR register is 0

If the value of the bank registers (DTB, ADB, USB, and SSB) is "00$_H$", the remainder after division is stored in the register of the instruction operands. Otherwise, the upper eight bits is specified by the bank register corresponding to the register of the instruction operand, and the lower 16 bits is the same as the address of the register of the instruction operand. The remainder is stored in the bank register specified by the upper eight bits.

Example:

If "DIV A,R0" is executed with DTB = "053_H" and RP = "03_H", the address of R0 is "0180_H" + RP ("03_H") × "10_H" + "08_H" (R0 corresponding address) = "$0001B8_H$". Since the data bank register (DTB) is specified by "DIV A,R0" as the bank register, the remainder is stored in address "$05301B8_H$", which was obtained by adding the bank address "053_H".

Note:

For information about the bank register and Ri and RWi registers, see "2.7 Registers".

■ Use of the "DIV A, Ri" and "DIVW A, RWi" Instructions without Precautions

To enable users to develop programs without having to take precautions for using the "DIV A,Ri" and "DIVW A,RWi" instructions, special compilers and assemblers are available. The special compiler does not generate the instructions in Table 2.11-1 . The special assemblers have a function that replaces the instructions in Table 2.11-1 with equivalent instruction strings. For the MB90360 series, use the following types of compilers and assemblers:

● Compiler

- cc907 V02L06 or later version, or fcc907s V30L02 or later version

● Assembler

- asm907a V03L04 or later version, or fasm907s V30L04 (Rev. 300004) or later version

CHAPTER 3

INTERRUPTS

This chapter explains the interrupts and function and operation of the extended intelligent I/O service in the MB90360 series.

3.1 Outline of Interrupts

The F²MC-16LX has interrupt functions that terminate the currently executing processing and transfer control to another specified program when a specified event occurs. There are four types of interrupt functions:

- **Hardware interrupt: Interrupt processing due to an internal resource event**
- **Software interrupt: Interrupt processing due to a software event occurrence instruction**
- **Extended intelligent I/O service (EI²OS): Transfer processing due to an internal resource event**
- **Exception: Termination due to an operation exception**

■ Hardware Interrupts

A hardware interrupt is activated by an interrupt request from an internal resource. A hardware interrupt request occurs when both the interrupt request flag and the interrupt enable flag in an internal resource are set. Therefore, an internal resource must have an interrupt request flag and interrupt enable flag to issue a hardware interrupt request.

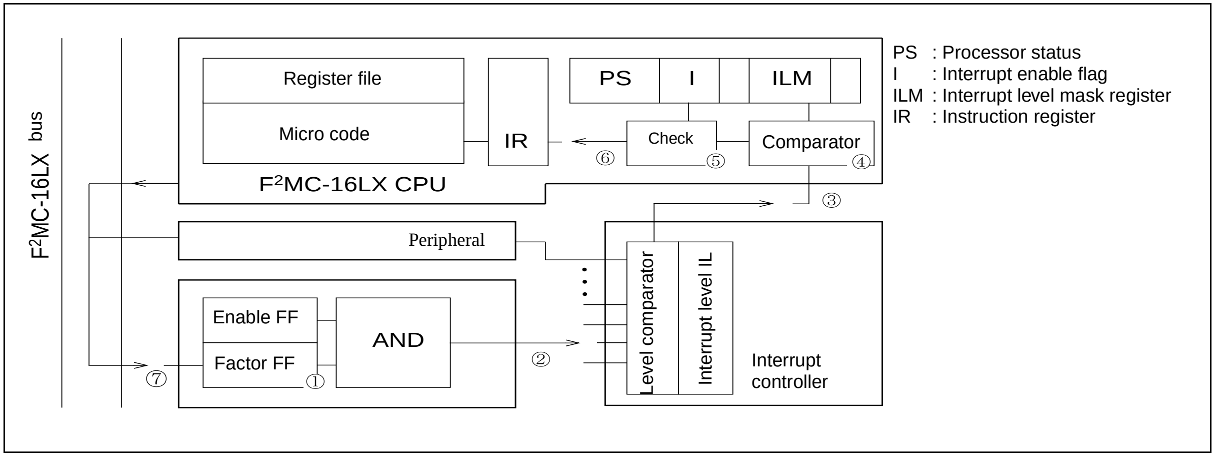
● Specifying an interrupt level

An interrupt level can be specified for the hardware interrupt. To specify an interrupt level, use the level setting bits (IL0, IL1, and IL2) of the interrupt controller.

● Masking a hardware interrupt request

A hardware interrupt request can be masked by using the I flag of the processor status register (PS) in the CPU and the ILM bits (IL0, IL1, and IL2). When an unmasked interrupt request occurs, the CPU saves 12 bytes of data that consists of registers PS, PC, PCB, DTB, ADB, DPR, and A in the memory area indicated by the SSB and SSP registers.

Figure 3.1-1 Overview of Hardware Interrupts

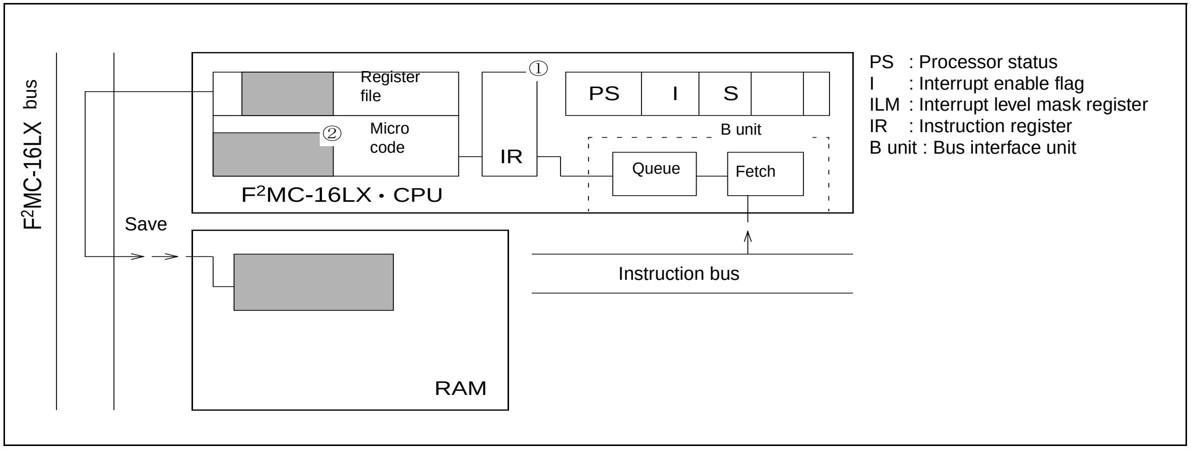


■ Software Interrupts

Interrupts requested by executing the INT instruction are software interrupts. An interrupt request by the INT instruction does not have an interrupt request or enable flag. An interrupt request is issued always by executing the INT instruction.

No interrupt level is assigned to the INT instruction. Therefore, ILM is not updated when the INT instruction is used. Instead, the I flag is cleared and the continuing interrupt requests are suspended.

Figure 3.1-2 Overview of Software Interrupts



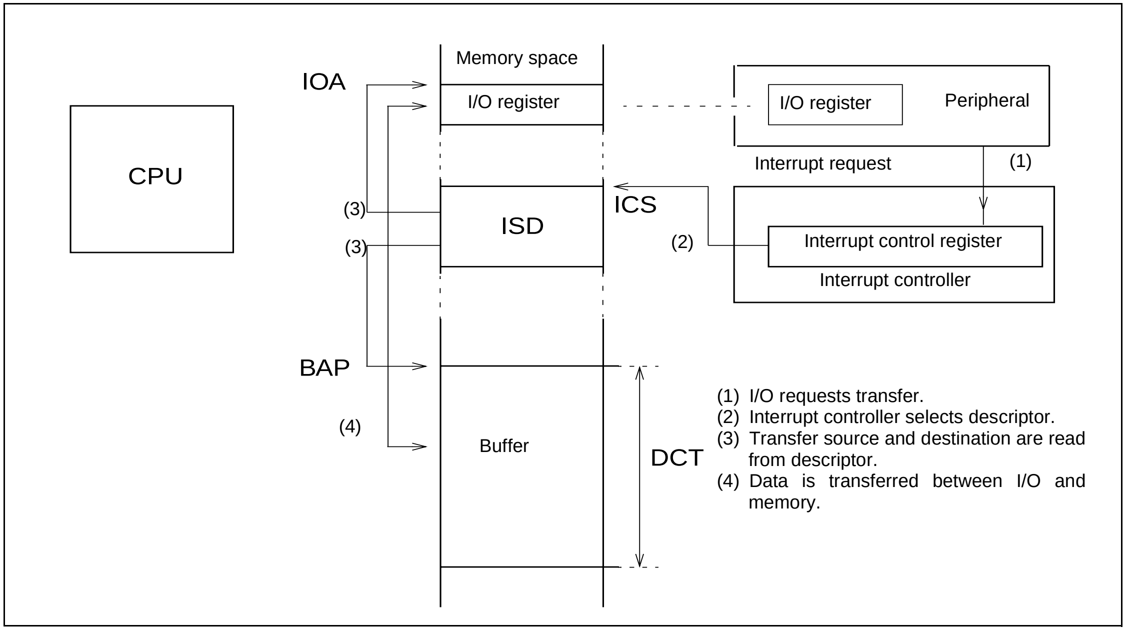
■ Extended Intelligent I/O Service (EI²OS)

The extended intelligent I/O service automatically transfers data between an internal resource and memory. This processing is traditionally performed by an interrupt processing program, but the EI²OS enables data to be transferred in a manner similar to a DMA (direct memory access) operation.

To activate the extended intelligent I/O service function from an internal resource, the interrupt control register (ICR) of the interrupt controller must have an extended intelligent I/O service enable flag (ISE).

The extended intelligent I/O service is started when an interrupt request occurs with 1 specified in the ISE flag. To generate a normal interrupt using a hardware interrupt request, set the ISE flag to 0.

Figure 3.1-3 Overview of the Extended Intelligent I/O Service (EI²OS)


CPU
IOA
Memory space
I/O register
I/O register
Peripheral
Interrupt request
(1)
(3)
(3)
ISD
ICS
(2)
Interrupt control register
Interrupt controller
BAP
(4)
Buffer
DCT
(1) I/O requests transfer.
(2) Interrupt controller selects descriptor.
(3) Transfer source and destination are read from descriptor.
(4) Data is transferred between I/O and memory.

■ Exceptions

Exception processing is basically the same as interrupt processing. When an exception is detected between instructions, ordinary processing is suspended, and exception processing is performed. In general, exception processing occurs as a result of an unexpected operation. Therefore, use exception processing for debugging programs or for activating recovery software in an emergency.

3.2 Interrupt Vector

An interrupt vector uses the same area for both hardware and software interrupts. For example, interrupt request number INT42 is used for a delayed hardware interrupt and for software interrupt INT #42. Therefore, the delayed interrupt and INT #42 call the same interrupt processing routine. Interrupt vectors are allocated between addresses FFFC00$_H$ and FFFFFF$_H$ as shown in Table 3.2-1 .

■ Interrupt Vector

Table 3.2-1 Interrupt Vector (1/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address lower	Vector address middle	Vector address upper	Mode register
		Number	Address				
INT 0 *	--	--	--	FFFFFC$_H$	FFFFFD$_H$	FFFFFE$_H$	Unused
INT 1 *	--	--	--	FFFFF8$_H$	FFFFF9$_H$	FFFFFA$_H$	Unused
. . .	--	--	--	. . .	. . .	. . .	. . .
INT 7 *	--	--	--	FFFFE0$_H$	FFFFE1$_H$	FFFFE2$_H$	Unused
INT 8	Reset	--	--	FFFFDC$_H$	FFFFDD$_H$	FFFFDE$_H$	FFFFDF$_H$
INT 9	INT9 instruction	--	--	FFFFD8$_H$	FFFFD9$_H$	FFFFDA$_H$	Unused
INT 10	Exception processing	--	--	FFFFD4$_H$	FFFFD5$_H$	FFFFD6$_H$	Unused
INT 11	Reserved	ICR00	0000B0$_H$	FFFFD0$_H$	FFFFD1$_H$	FFFFD2$_H$	Unused
INT 12	Reserved			FFFFCC$_H$	FFFFCD$_H$	FFFFCE$_H$	Unused
INT 13	CAN1 reception	ICR01	0000B1$_H$	FFFFC8$_H$	FFFFC9$_H$	FFFFCA$_H$	Unused
INT 14	CAN1 transmission/ node status			FFFFC4$_H$	FFFFC5$_H$	FFFFC6$_H$	Unused
INT 15	Reserved	ICR02	0000B2$_H$	FFFFC0$_H$	FFFFC1$_H$	FFFFC2$_H$	Unused
INT 16	Reserved			FFFFBC$_H$	FFFFBD$_H$	FFFFBE$_H$	Unused
INT 17	Reserved	ICR03	0000B3$_H$	FFFFB8$_H$	FFFFB9$_H$	FFFFBA$_H$	Unused
INT 18	Reserved			FFFFB4$_H$	FFFFB5$_H$	FFFFB6$_H$	Unused
INT 19	16-bit reload timer 2	ICR04	0000B4$_H$	FFFFB0$_H$	FFFFB1$_H$	FFFFB2$_H$	Unused
INT 20	16-bit reload timer 3			FFFFAC$_H$	FFFFAD$_H$	FFFFAE$_H$	Unused
INT 21	Reserved	ICR05	0000B5$_H$	FFFFA8$_H$	FFFFA9$_H$	FFFFAA$_H$	Unused
INT 22	Reserved			FFFFA4$_H$	FFFFA5$_H$	FFFFA6$_H$	Unused
INT 23	PPG C/D	ICR06	0000B6$_H$	FFFFA0$_H$	FFFFA1$_H$	FFFFA2$_H$	Unused
INT 24	PPG E/F			FFFF9C$_H$	FFFF9D$_H$	FFFF9E$_H$	Unused

Table 3.2-1 Interrupt Vector (2/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address lower	Vector address middle	Vector address upper	Mode register
		Number	Address				
INT 25	Timebase timer 3	ICR07	0000B7$_{H}$	FFFF98$_{H}$	FFFF99$_{H}$	FFFF9A$_{H}$	Unused
INT 26	External interrupt 8 to 11			FFFF94$_{H}$	FFFF95$_{H}$	FFFF96$_{H}$	Unused
INT 27	Watch timer	ICR08	0000B8$_{H}$	FFFF90$_{H}$	FFFF91$_{H}$	FFFF92$_{H}$	Unused
INT 28	External interrupt 12 to 15			FFFF8C$_{H}$	FFFF8D$_{H}$	FFFF8E$_{H}$	Unused
INT 29	A/D converter	ICR09	0000B9$_{H}$	FFFF88$_{H}$	FFFF89$_{H}$	FFFF8A$_{H}$	Unused
INT 30	I/O timer 0			FFFF84$_{H}$	FFFF85$_{H}$	FFFF86$_{H}$	Unused
INT 31	Reserved	ICR10	0000BA$_{H}$	FFFF80$_{H}$	FFFF81$_{H}$	FFFF82$_{H}$	Unused
INT 32	Reserved			FFFF7C$_{H}$	FFFF7D$_{H}$	FFFF7E$_{H}$	Unused
INT 33	Input capture 0 to 3	ICR11	0000BB$_{H}$	FFFF78$_{H}$	FFFF79$_{H}$	FFFF7A$_{H}$	Unused
INT 34	Reserved			FFFF74$_{H}$	FFFF75$_{H}$	FFFF76$_{H}$	Unused
INT 35	UART 0 reception	ICR12	0000BC$_{H}$	FFFF70$_{H}$	FFFF71$_{H}$	FFFF72$_{H}$	Unused
INT 36	UART 0 transmission			FFFF6C$_{H}$	FFFF6D$_{H}$	FFFF6E$_{H}$	Unused
INT 37	UART 1 reception	ICR13	0000BD$_{H}$	FFFF68$_{H}$	FFFF69$_{H}$	FFFF6A$_{H}$	Unused
INT 38	UART 1 transmission			FFFF64$_{H}$	FFFF65$_{H}$	FFFF66$_{H}$	Unused
INT 39	Reserved	ICR14	0000BE$_{H}$	FFFF60$_{H}$	FFFF61$_{H}$	FFFF62$_{H}$	Unused
INT 40	Reserved			FFFF5C$_{H}$	FFFF5D$_{H}$	FFFF5E$_{H}$	Unused
INT 41	Flash memory	ICR15	0000BF$_{H}$	FFFF58$_{H}$	FFFF59$_{H}$	FFFF5A$_{H}$	Unused
INT 42	Delayed interrupt generation module			FFFF54$_{H}$	FFFF55$_{H}$	FFFF56$_{H}$	Unused
INT 43	--	--	--	FFFF50$_{H}$	FFFF51$_{H}$	FFFF52$_{H}$	Unused
. . .	--	--	--	. . .	. . .	. . .	. . .
INT 254	--	--	--	FFFC04$_{H}$	FFFC05$_{H}$	FFFC06$_{H}$	Unused
INT 255	--	--	--	FFFC00$_{H}$	FFFC01$_{H}$	FFFC02$_{H}$	Unused

*: When PCB is FF$_{H}$, the vector area for the CALLV instruction overlaps that for INT #vct8 (#0 to #7). Care must be taken when using the CALLV instruction.

3.3 Interrupt Control Registers (ICR)

The interrupt control registers are in the interrupt controller. Each interrupt control register has a corresponding I/O that has an interrupt function. The interrupt control registers have the following 3 functions:

- **Setting an interrupt level for corresponding peripherals**
- **Selecting whether to use an ordinary interrupt or extended intelligent I/O service for the corresponding peripherals**
- **Selecting the extended intelligent I/O service channel**

Do not access an interrupt control register by using a read-modify-write instruction, as doing so causes a misoperation.

■ Interrupt Control Register (ICR)

Figure 3.3-1 is a diagram of the bit configuration of an interrupt control register.

Figure 3.3-1 Interrupt Control Register (ICR)

15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
ICS3	ICS2	ICS1 or S1	ICS0 or S0	ISE	IL2	IL1	IL0	Interrupt control register 00000111_B when reset
W	W	*	*	R/W	R/W	R/W	R/W	

*: '1' is read always.
ICS1 and ICS0 are valid for write only. S1 and S0 are valid for read only.

Note:

ICS3 to ICS0 are valid only when EI^2OS is activated. Set '1' in ISE to activate EI^2OS, and set '0' in ISE not to activate it. When EI^2OS is not to be activated, any value can be set in ICS3 to ICS0.

[bit 10 to bit 8, bit 2 to bit 0] IL0, IL1, and IL2 (interrupt level setting bits)

These bits are readable and writable and specify the interrupt level of the corresponding internal resources. Upon a reset, these bits are initialized to level 7 (no interrupt). Table 3.3-1 describes the relationship between the interrupt level setting bits and interrupt levels.

Table 3.3-1 Interrupt Level Setting Bits and Interrupt Levels

ILM2	ILM1	ILM0	Level
0	0	0	0 (strongest)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (weakest)
1	1	1	7 (no interrupt)

[bit 11, bit 3] ISE (extended intelligent I/O service enable bits)

The ISE bit is readable and writable. In response to an interrupt request, EI^{2}OS is activated when '1' is set in the ISE bit and an interrupt sequence is activated when '0' is set in the ISE bit. Upon completion of EI^{2}OS, the ISE bit is cleared to a zero. If the corresponding peripheral does not have the EI^{2}OS function, the ISE bit must be set to '0' on the software side.

Upon a reset, the ISE bit is initialized to '0'.

[bit 15 to bit 12, bit 7 to bit 4] ICS 3 to ICS 0 (extended intelligent I/O service channel select bits)

ICS3 to ICS0 are write-only bits. These bits specify the EI^{2}OS channel. The values set in these bits determined the extended intelligent I/O service descriptor addresses in memory, which is explained later. The ICS bits are initialized to "0000$_B$" by a reset.

Table 3.3-2 describes the correspondence between the ICS bits, channel numbers, and descriptor addresses.

Table 3.3-2 ICS Bits, Channel Numbers, and Descriptor Address

ICS3	ICS2	ICS1	ICS0	Selected channel	Descriptor address
0	0	0	0	0	000100$_H$
0	0	0	1	1	000108$_H$
0	0	1	0	2	000110$_H$
0	0	1	1	3	000118$_H$
0	1	0	0	4	000120$_H$
0	1	0	1	5	000128$_H$
0	1	1	0	6	000130$_H$
0	1	1	1	7	000138$_H$
1	0	0	0	8	000140$_H$
1	0	0	1	9	000148$_H$
1	0	1	0	10	000150$_H$
1	0	1	1	11	000158$_H$
1	1	0	0	12	000160$_H$
1	1	0	1	13	000168$_H$
1	1	1	0	14	000170$_H$
1	1	1	1	15	000178$_H$

[bit 13, bit 12, bits 5, bit 4] S0 and S1 (extended intelligent I/O service status)

S0 and S1 are read-only bits. The values set in these bits indicate the end condition of EI^{2}OS. These bits are initialized to '00' upon a reset.

Table 3.3-3 shows the relationship between the S bits and the end conditions.

Table 3.3-3 S Bits and End Conditions

S1	S0	End conditions
0	0	EI^{2}OS running or not activated
0	1	Stop status by count end
1	0	Reserved
1	1	Stop status by request from internal resource

3.4 Interrupt Flow

Figure 3.4-1 shows the interrupt flow.

■ Interrupt Flow

Figure 3.4-1 Interrupt Flow

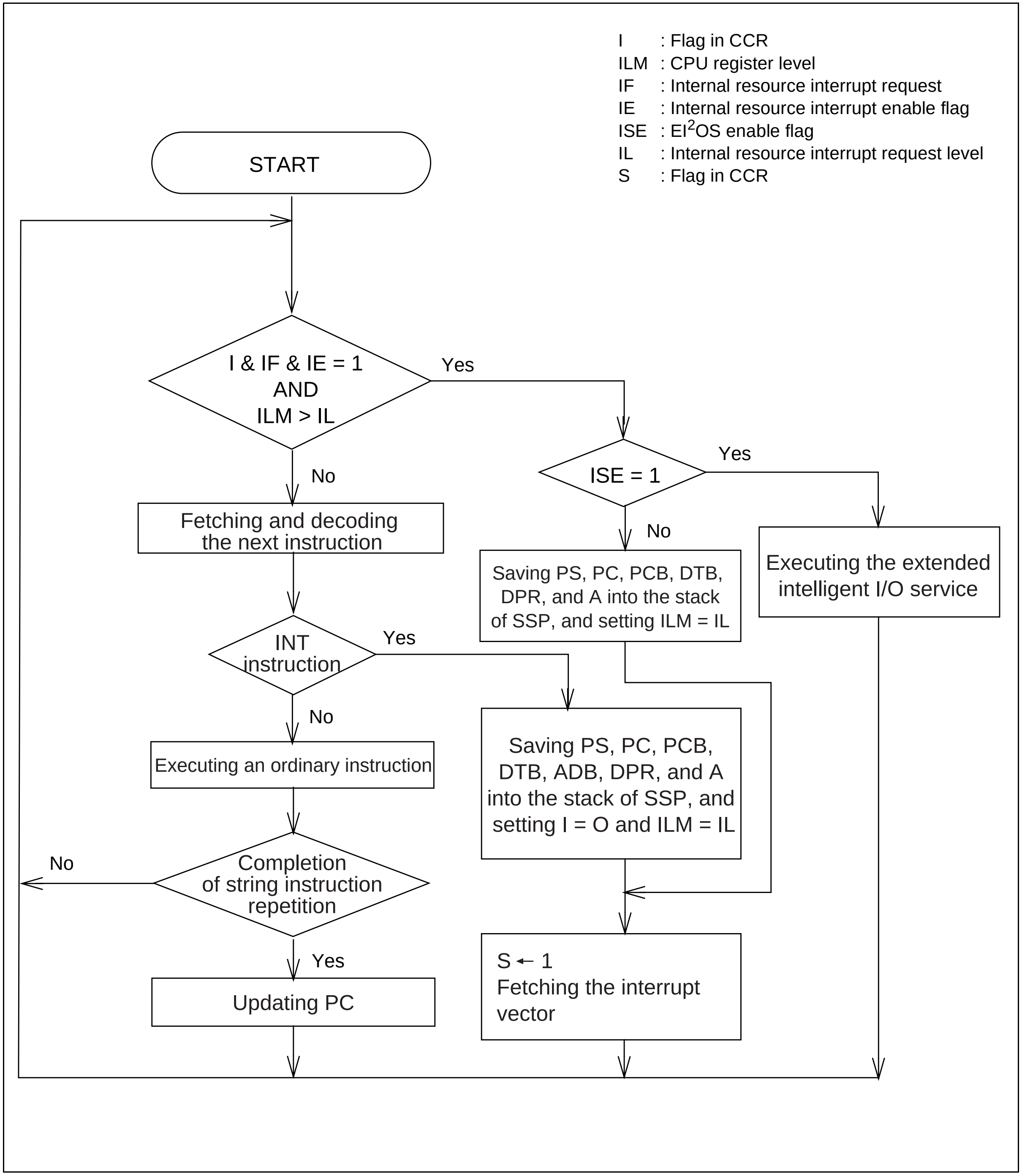
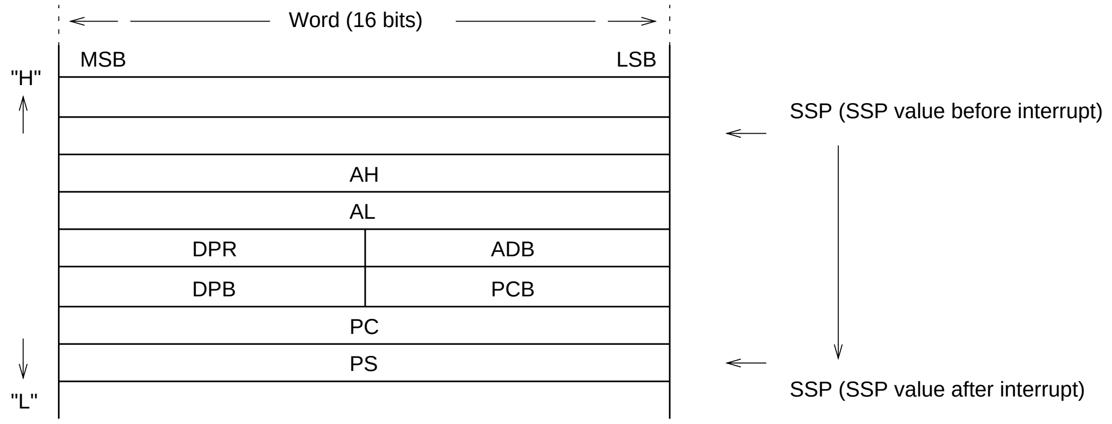

I : Flag in CCR
ILM : CPU register level
IF : Internal resource interrupt request
IE : Internal resource interrupt enable flag
ISE : EI2OS enable flag
IL : Internal resource interrupt request level
S : Flag in CCR
START
I & IF & IE = 1 AND ILM > IL
Yes
No
ISE = 1
Yes
No
Fetching and decoding the next instruction
Saving PS, PC, PCB, DTB, DPR, and A into the stack of SSP, and setting ILM = IL
Executing the extended intelligent I/O service
INT instruction
Yes
No
Executing an ordinary instruction
Saving PS, PC, PCB, DTB, ADB, DPR, and A into the stack of SSP, and setting I = O and ILM = IL
No
Completion of string instruction repetition
Yes
Updating PC
S ← 1
Fetching the interrupt vector

Figure 3.4-2 Register Saving during Interrupt Processing


Word (16 bits)
MSB
LSB
"H"
"L"
AH
AL
DPR
ADB
DPB
PCB
PC
PS
SSP (SSP value before interrupt)
SSP (SSP value after interrupt)

3.5 Hardware Interrupts

In response to an interrupt request signal from an internal resource, the CPU pauses current program execution and transfers control to the interrupt processing program defined by the user. This function is called the hardware interrupt function.

■ Hardware Interrupts

A hardware interrupt occurs when the relevant conditions are satisfied as a result of two operations: comparison between the interrupt request level and the value in the interrupt level mask register (ILM) of PS in the CPU, and hardware reference to the I flag value of PS.

The CPU performs the following processing when a hardware interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- Sets ILM in the PS register. The currently requested interrupt level is automatically set.
- Fetches the corresponding interrupt vector value and branches to the processing indicated by that value.

■ Structure of Hardware Interrupt

Hardware interrupts are handled by the following 3 sections:

● Internal resources

Interrupt enable and request bits: Used to control interrupt requests from resources.

● Interrupt controller

ICR: Assigns interrupt levels and determines the priority levels of simultaneously requested interrupts.

● CPU

I and ILM: Used to compare the requested and current interrupt levels and to identify the interrupt enable status.

Microcode: Interrupt processing step

The status of these sections are indicated by the resource control registers for internal resources, the ICR for the interrupt controller, and the CCR value for the CPU. To use a hardware interrupt, set the three sections beforehand by using software.

The interrupt vector table referred during interrupt processing is assigned to addresses FFFC00$_H$ to FFFFFF$_H$ in memory. These addresses are shared with software interrupts.

"Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers" in "APPENDIX D List of Interrupt Vectors" shows the assignment of the MB90360 series.

3.5.1 Hardware Interrupt Operation

An internal resource that has the hardware interrupt request function has an interrupt request flag and interrupt enable flag. The interrupt request flag indicates whether an interrupt request exists, and the interrupt enable flag indicates whether the relevant internal resource requests an interrupt to the CPU. The interrupt request flag is set when an event that is unique to the internal resource occurs. When the interrupt enable flag indicates "enable", the resource issues an interrupt request to the interrupt controller.

■ Hardware Interrupt Operation

When two or more interrupt requests are received at the same time, the interrupt controller compares the interrupt levels (IL) in ICR, selects the request at the highest level (the smallest IL value), then reports that request to the CPU. If multiple requests are at the same level, the interrupt controller selects the request with the lowest interrupt number. The relationship between the interrupt requests and ICRs is determined by the hardware.

The CPU compares the received interrupt level (IL) and the ILM in the PS register. If the interrupt level is smaller than the ILM value and the I bit of the PS register is set to '1', the CPU activates the interrupt processing microcode after the currently executing instruction is completed. The CPU refers the ISE bit of the ICR of the interrupt controller at the beginning of the interrupt processing microcode, checks that the ISE bit is 0 (interrupt), and activates the interrupt processing body.

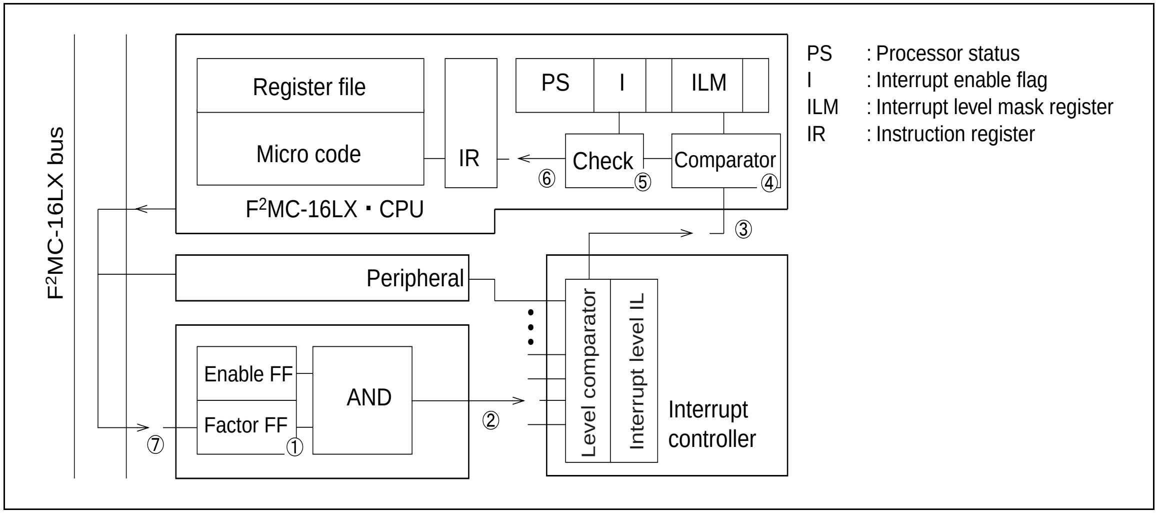
The interrupt processing body saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP, fetches 3 bytes of interrupt vector, loads them onto PC and PCB, updates the ILM of PS to a level value of the received interrupt request, sets the S flag, then performs branch processing. As a result, the interrupt processing program defined by the user is executed next.

3.5.2 Occurrence and Release of Hardware Interrupt

Figure 3.5-1 shows the processing flow from occurrence of a hardware interrupt to release of the interrupt request in an interrupt processing program.

■ Occurrence and Release of Hardware Interrupt

Figure 3.5-1 Occurrence and Release of Hardware Interrupt



1. An interrupt cause occurs in a peripheral.
2. The interrupt enable bit in the peripheral is referred. If interrupts are enabled, the peripheral issues an interrupt request to the interrupt controller.
3. Upon reception of the interrupt request, the interrupt controller determines the priority levels of simultaneously requested interrupts. Then, the interrupt controller transfers the interrupt level of the corresponding interrupt to the CPU.
4. The CPU compares the interrupt level requested by the interrupt controller with the ILM bit of the processor status register.
5. If the comparison shows that the requested level is higher than the current interrupt processing level, the I flag value of the same processor status register is checked.
6. If the check in step 5. shows that the I flag indicates interrupt enable status, the requested level is written to the ILM bit. Interrupt processing is performed as soon as the currently executing instruction is completed, then control is transferred to the interrupt processing routine.
7. When the interrupt cause of step 1. is cleared by software in the user interrupt processing routine, the interrupt request is completed.

The time required for the CPU to execute the interrupt processing in steps 6. and 7. is shown below.

See Table 3.5-1 for the cycle count compensation value.

Interrupt start: 24 + 6 × (Table 3.3-2 machine cycles)

Interrupt return: 15 + 6 × (Table 3.3-2 machine cycles)   RETI instruction

Table 3.5-1 Compensation Values for Interrupt Processing Cycle Count

Address indicated by stack pointer	Cycle count compensation value
Internal area, even-number address	0
Internal area, add-number address	+2

3.5.3 Multiple interrupts

As a special case, no hardware interrupt request can be accepted while data is being written to the I/O area. This is intended to prevent the CPU from operating falsely because of an interrupt request issued while an interrupt control register for a resource is being updated.

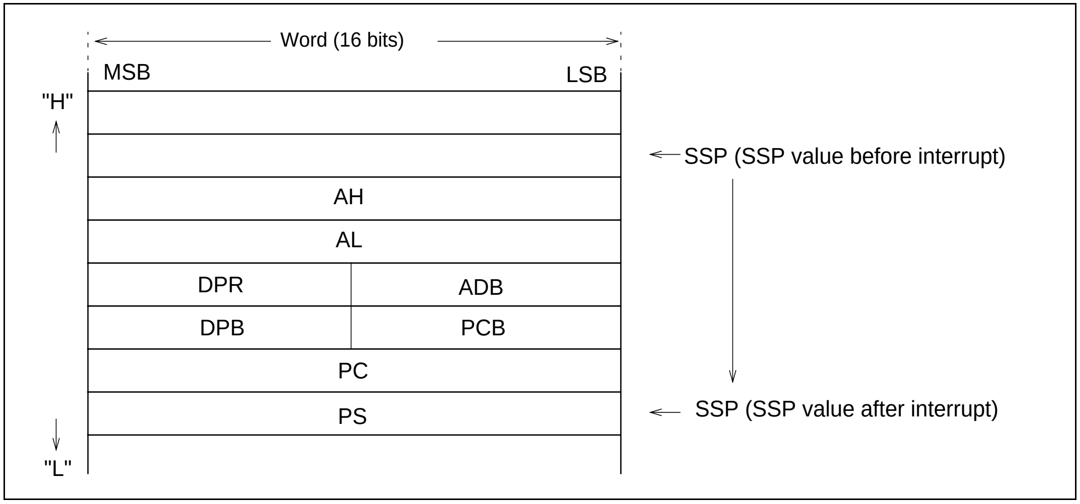
If an interrupt occurs during interrupt processing, a higher-level interrupt is processed first.

■ Multiple Interrupts

The F^2MC-16LX CPU supports multiple interrupts. If an interrupt of a higher level occurs while another interrupt is being processed, control is transferred to the high-level interrupt after the currently executing instruction is completed. After processing of the high-level interrupt is completed, the original interrupt processing is resumed. An interrupt of the same or lower level may be generated while another interrupt is being processed. If this happens, the new interrupt request is suspended until the current interrupt processing is completed, unless the ILM value or I flag is changed by an instruction. The extended intelligent I/O service cannot be activated from multiple sources; while an extended intelligent I/O service is being processed, all other interrupt requests or extended intelligent I/O service requests are suspended.

Figure 3.5-2 shows the order of the registers saved in the stack.

Figure 3.5-2 Registers Saved in Stack


Word (16 bits)
MSB
LSB
"H"
SSP (SSP value before interrupt)
AH
AL
DPR
ADB
DPB
PCB
PC
PS
SSP (SSP value after interrupt)
"L"

3.6 Software Interrupts

In response to execution of a special instruction, control is transferred from the program currently executed by the CPU to the interrupt processing program defined by the user. This is called the software interrupt function. A software interrupt occurs always when the software interrupt instruction is executed.

■ Software Interrupts

The CPU performs the following processing when a software interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- Sets I in the PS register. Interrupts are automatically disabled.
- Fetches the corresponding interrupt vector value, then branches to the processing indicated by that value.

A software interrupt request issued by the INT instruction has no interrupt request or enable flag. A software interrupt request is always issued by executing the INT instruction.

The INT instruction does not have an interrupt level. Therefore, the INT instruction does not update ILM. The INT instruction clears the I flag to suspend subsequent interrupt requests.

■ Structure of Software Interrupts

Software interrupts are handled within the CPU:

CPU.....Microcode: Interrupt processing step

■ List of Interrupt Vectors

"Table D-1 Interrupt Vectors" in APPENDIX D lists the interrupt vectors of the MB90360 series.

Software interrupts share the same interrupt vector area with hardware interrupts.

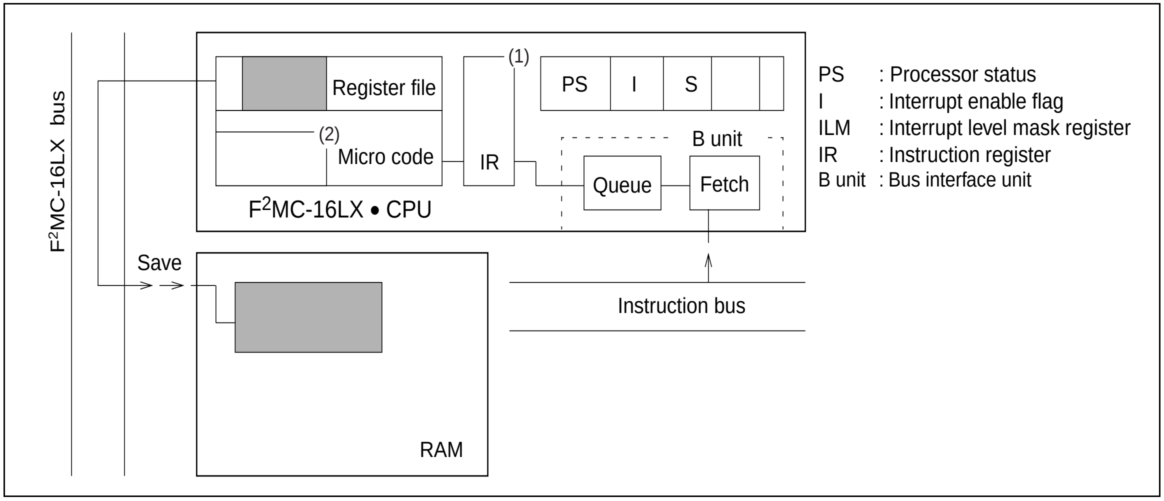
For example, interrupt request number INT 12 is used for external interrupt #0 to #7 of a hardware interrupt as well as for INT #12 of a software interrupt. Therefore, external interrupt #0 and INT #12 call the same interrupt processing routine.

■ Software Interrupt Operation

When the CPU fetches and executes the software interrupt instruction, the software interrupt processing microcode is activated. The software interrupt processing microcode saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP. The microcode then fetches 3 bytes of interrupt vector and loads them onto PC and PCB, resets the I flag, and sets the S flag. Then, the microcode performs branch processing. As a result, the interrupt processing program defined by the user application program is executed next.

Figure 3.6-1 illustrates the flow from the occurrence of a software interrupt until there is no interrupt request in the interrupt processing program.

Figure 3.6-1 Occurrence and Release of Software Interrupt



(1) The software interrupt instruction is executed.

(2) Special CPU registers in the register file are saved according to the microcode corresponding to the software interrupt instruction.

(3) The interrupt processing is completed with the RETI instruction in the user interrupt processing routine.

■ Others

When the program bank register (PCB) is FF_H, the CALLV instruction vector area overlaps the table of the INT #vct8 instruction. When designing software, ensure that the CALLV instruction does not use the same address as that of the #vct8 instruction.

"Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers" in APPENDIX D shows the relationship of interrupt cause, interrupt vector, and interrupt control register in the MB90360 series.

3.7 Extended Intelligent I/O Service (EI^{2}OS)

The EI^{2}OS function, a kind of hardware interrupt operation, automatically transfers data between input and output and memory. An interrupt processing program was conventionally used for such processing, but EI^{2}OS enables data transfer to be performed like DMA (direct memory access).

■ Extended Intelligent I/O Service (EI^{2}OS)

EI^{2}OS has the following advantages over the conventional method:

- The program size can be small because it is not necessary to write a transfer program.
- No internal register is used for transfer, eliminating the need for register saving and increasing the transfer speed.
- Transfer can be terminated from I/O, preventing unnecessary data from being transferred.
- The buffer address may either be incremented or left unupdated.
- The I/O register address may either be incremented or left unupdated (buffer address is update).

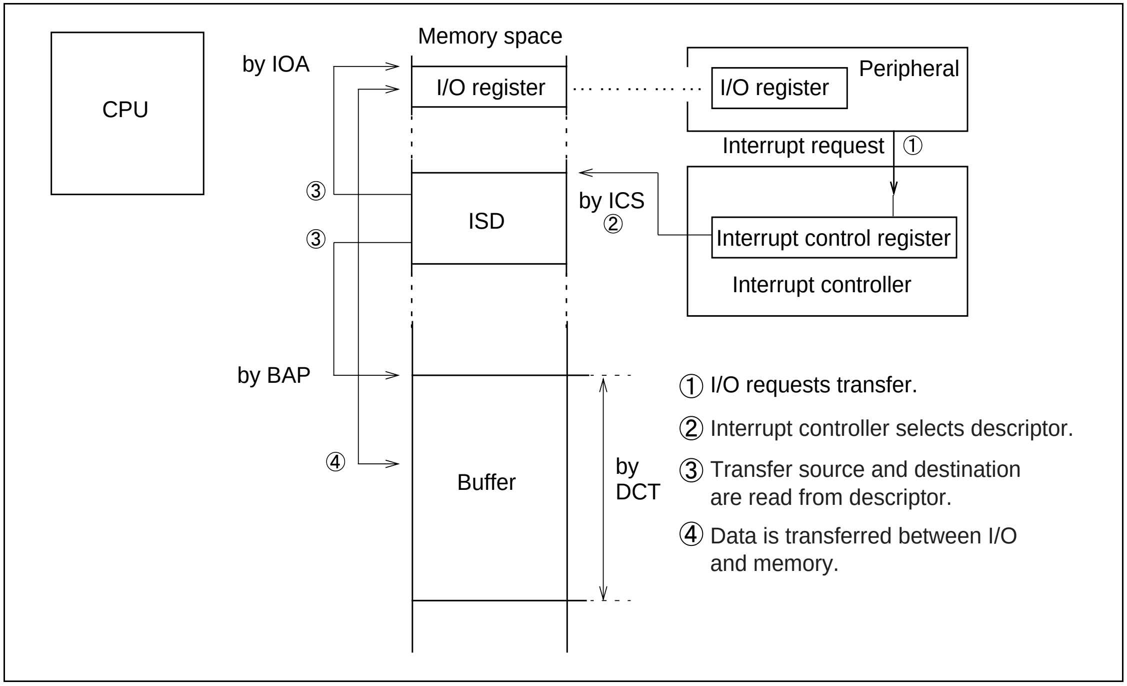
At the end of EI^{2}OS, processing automatically branches to an interrupt processing routine after the end condition is set. Thus, the user can identify the end condition.

To implement EI^{2}OS, the hardware is distributed in two blocks. Each block has the following registers and descriptors.

- Interrupt control register: Exists in the interrupt controller and indicates the ISD address.
- Extended intelligent I/O service descriptor (ISD): Exists in RAM and holds the transfer mode, I/O address, number of transfers, and buffer address.

Figure 3.7-1 outlines the extended intelligent I/O service.

Figure 3.7-1 Outline of Extended Intelligent I/O Service


CPU
Memory space
by IOA
I/O register
I/O register
Peripheral
Interrupt request ①
③
ISD
by ICS
②
③
Interrupt control register
Interrupt controller
by BAP
④
Buffer
by DCT
① I/O requests transfer.
② Interrupt controller selects descriptor.
③ Transfer source and destination are read from descriptor.
④ Data is transferred between I/O and memory.

Note:

- The area that can be specified by IOA is between 000000_H and $00FFFF_H$.
- The area that can be specified by BAP is between 000000_H and $FFFFFF_H$.
- The maximum transfer count that can be specified by DCT is 65,536.

■ Structure

EI^{2}OS is handled by the following 4 sections:

Internal resources

Interrupt enable and request bits: Used to control interrupt requests from resources.

Interrupt controller

ICR: Assigns interrupt levels, determines the priority levels of simultaneously requested interrupts, and selects the EI^{2}OS operation.

CPU

I and ILM: Used to compare the requested and current interrupt levels and to identify the interrupt enable status

Microcode: EI^{2}OS processing step

RAM

Descriptor: Describes the EI^{2}OS transfer information.

3.7.1 Extended Intelligent I/O Service Descriptor (ISD)

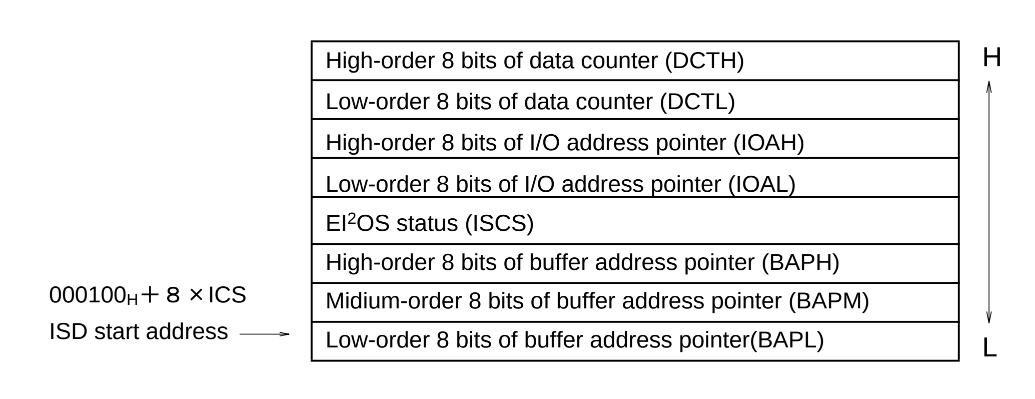
The extended intelligent I/O service descriptor exists between 000100$_H$ and 00017F$_H$ in internal RAM and consists of the following items:

- **Data transfer control data**
- **Status data**
- **Buffer address pointer**

■ Extended Intelligent I/O Service Descriptor (ISD)

Figure 3.7-2 shows the configuration of the extended intelligent I/O service descriptor.

Figure 3.7-2 Extended Intelligent I/O Service Descriptor Configuration



High-order 8 bits of data counter (DCTH)

Low-order 8 bits of data counter (DCTL)

High-order 8 bits of I/O address pointer (IOAH)

Low-order 8 bits of I/O address pointer (IOAL)

EI^{2}OS status (ISCS)

High-order 8 bits of buffer address pointer (BAPH)

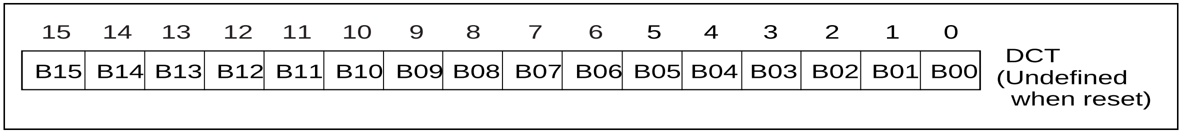
Midium-order 8 bits of buffer address pointer (BAPM)

Low-order 8 bits of buffer address pointer(BAPL)

■ Data Counter (DCT)

This is a 16-bit register that works as a counter corresponding to the number of data items transferred. This counter is decremented by one before data transfer. EI^{2}OS is terminated when this counter reaches 0. Figure 3.7-3 is a diagram of the data counter configuration.

Figure 3.7-3 Data Counter Configuration



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
B15	B14	B13	B12	B11	B10	B09	B08	B07	B06	B05	B04	B03	B02	B01	B00	DCT (Undefined when reset)

■ I/O register address pointer (IOA)

This is a 16-bit register that indicates the low-order address (A15 to A0) of the buffer and I/O register used for data transfer. The high-order address (A23 to A16) are all zeroes, and any I/O between addresses 000000_H and $00FFFF_H$ can be specified. Figure 3.7-4 is a diagram of the IOA configuration.

Figure 3.7-4 I/O Register Address Pointer Configuration

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
A15	A14	A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00	IOA (Undefined when reset)

■ Buffer Address Pointer (BAP)

This 24-bit register holds the address used for the next EI^2OS transfer. BAP exists for each EI^2OS channel. Therefore, each EI^2OS channel can be used for transfer with anywhere in the 16M bytes space. If the BF bit of ISCS is set to '0' (update enabled), only the low-order 16 bits of BAP changes and BAPH does not change.

3.7.2 EI^{2}OS Status Register (ISCS)

This eight-bit register indicates the update direction (increment/decrement), transfer data format (byte/word), and transfer direction of the buffer address pointer and the I/O register address pointer. This register also indicates whether the buffer address pointer or I/O register address pointer is updated or fixed.

■ EI^{2}OS Status Register (ISCS)

Figure 3.7-5 is a diagram of the ISCS configuration.
Be sure to write "0" in bit 7 to bit 5 of ISCS.

Figure 3.7-5 ISCS Configuration

7	6	5	4	3	2	1	0	
Reserved	Reserved	Reserved	IF	BW	BF	DIR	SE	ISCS (Undefined when reset)

Each bit is described below.

[bit 4] IF: Specify whether the I/O register address pointer is updated or fixed.

0: The I/O register address pointer is updated after data transfer.

1: The I/O register address pointer is not updated after data transfer.

Note:
Only increment is allowed.

[bit 3] BW: Specify the transfer data length.

0: Byte

1: Word

[bit 2] BF: Specify whether the buffer address pointer is updated or fixed.

0: The buffer address pointer is updated after data transfer.

1: The buffer address pointer is not updated after data transfer.

Note:
Only the low-order 16 bits of the buffer address pointer are updated. Only increment is allowed.

[bit 1] DIR: Specify the data transfer direction.

0: I/O --> Buffer

1: Buffer --> I/O

[bit 0] SE: Control the termination of the extended intelligent I/O service based on internal resource requests.

0: The extended intelligent I/O service is not terminated by a internal resource request.

1: The extended intelligent I/O service is terminated by a internal resource request.

3.8 Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI^{2}OS)

Figure 3.8-1 is a diagram of the EI^{2}OS operation flow. Figure 3.8-2 is a diagram of the EI^{2}OS use procedure.

■ EI^{2}OS Operation Flow

Figure 3.8-1 EI^{2}OS Operation Flow

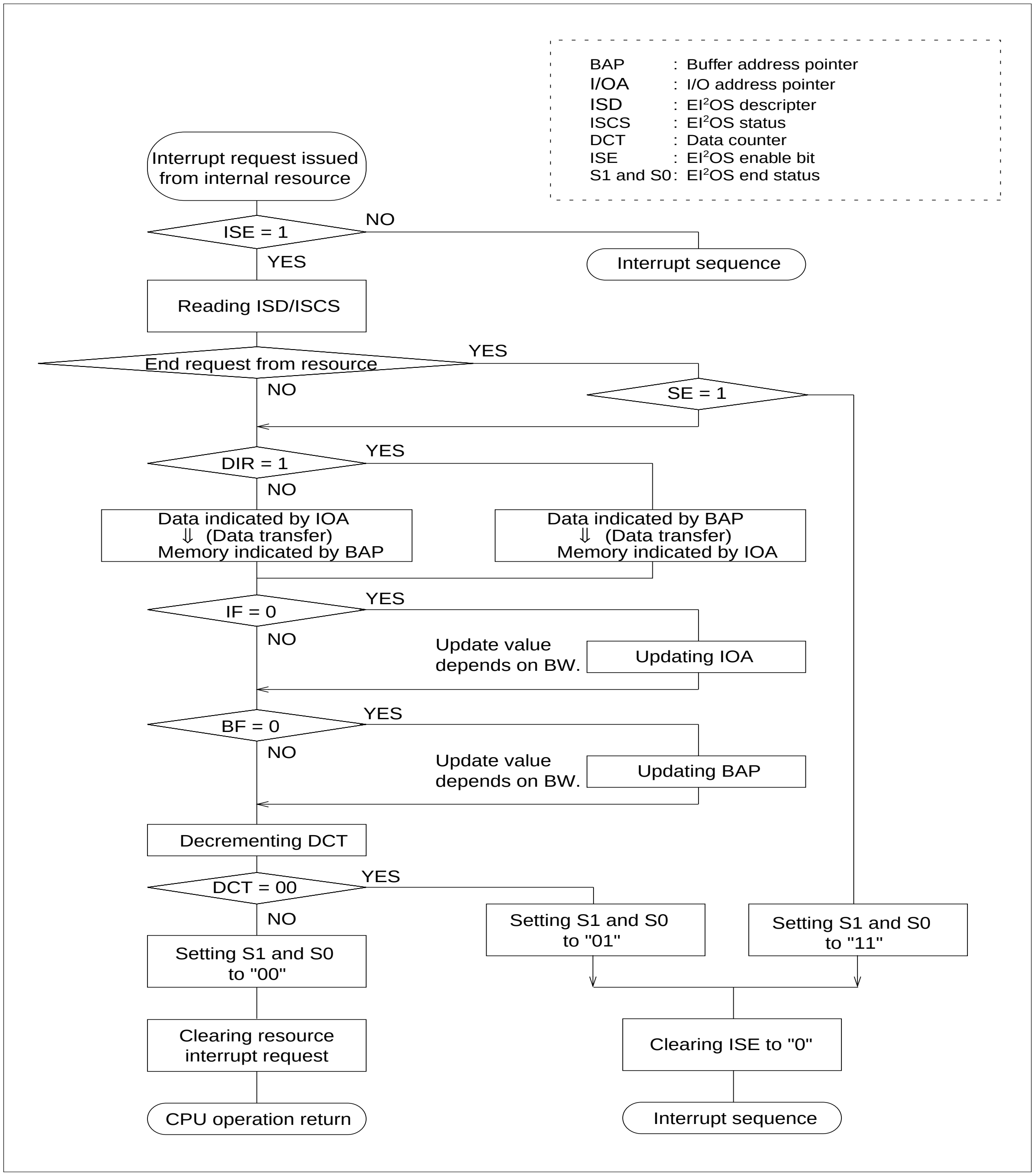
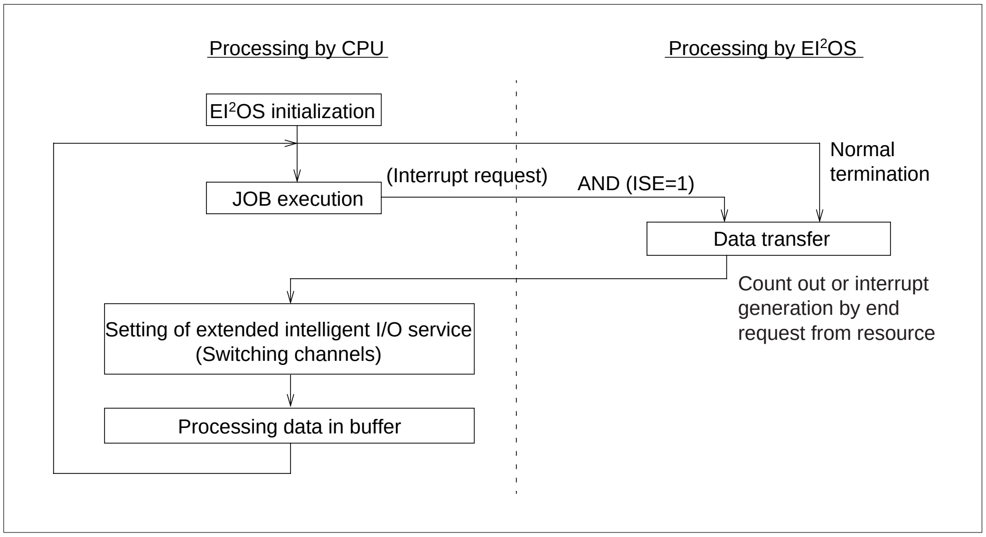

BAP : Buffer address pointer
I/OA : I/O address pointer
ISD : EI²OS descripter
ISCS : EI²OS status
DCT : Data counter
ISE : EI²OS enable bit
S1 and S0 : EI²OS end status
Interrupt request issued from internal resource
ISE = 1
NO
YES
Interrupt sequence
Reading ISD/ISCS
End request from resource
YES
NO
SE = 1
DIR = 1
YES
NO
Data indicated by IOA ⇓ (Data transfer) Memory indicated by BAP
Data indicated by BAP ⇓ (Data transfer) Memory indicated by IOA
IF = 0
YES
NO
Update value depends on BW.
Updating IOA
BF = 0
YES
NO
Update value depends on BW.
Updating BAP
Decrementing DCT
DCT = 00
YES
NO
Setting S1 and S0 to "00"
Setting S1 and S0 to "01"
Setting S1 and S0 to "11"
Clearing resource interrupt request
Clearing ISE to "0"
CPU operation return
Interrupt sequence

Figure 3.8-2 EI^{2}OS Use Flow


Processing by CPU
Processing by EI2OS
EI2OS initialization
Normal termination
(Interrupt request)
AND (ISE=1)
JOB execution
Data transfer
Count out or interrupt generation by end request from resource
Setting of extended intelligent I/O service (Switching channels)
Processing data in buffer

The extended EI^{2}OS execution time for each flow is described below.

- When data transfer continues (when the stop condition is not satisfied)

  (Figure 3.8-1 + Table 3.8-2) machine cycles

- When a stop request is issued from a resource

  (36 + 6 × Table 3.5-1) machine cycles

- When the counting is completed

  (Table 3.8-1 + Table 3.8-2 + (21 + 6 × Table 3.5-1) machine cycles

Table 3.8-1 Execution Time when the EI^{2}OS Continues

ISCS SE bit		Set to "0"		Set to "1"	
I/O address pointer		Fixed	Updated	Fixed	Updated
Buffer address pointer	Fixed	32	34	33	35
	Updated	34	36	35	37

Table 3.8-2 Data Transfer Compensation Values for EI^{2}OS Execution Time

I/O address pointer			Internal access	
			B/E	O
Buffer address pointer	Internal access	B/E	0	+2
		O	+2	+4

B: Byte data transfer
E: Even address word transfer
O: Odd address word transfer

Table 3.8-3 Interrupt Handling Time

Address pointed to by the stack pointer	Compensation value [cycle]
External 8 bits	+4
External even-numbered address	+1
External odd-numbered address	+4
Internal even-numbered address	0
Internal odd-numbered address	+2

3.9 Exceptions

The F^{2}MC-16LX performs exception processing when the following event occurs:

■ Execution of an Undefined Instruction

Exception processing is fundamentally the same as interrupt processing. When an exception is detected between instructions, exception processing is performed separately from ordinary processing. In general, exception processing is performed as a result of an unexpected operation. Fujitsu recommends using exception processing for debugging or for activating emergency recovery software.

■ Exception Due to Execution of an Undefined Instruction

The F^{2}MC-16LX handles all codes that are not defined in the instruction map as undefined instructions. When an undefined instruction is executed, processing equivalent to the INT 10 software interrupt instruction is performed. Specifically, the AL, AH, DPR, DTB, ADB, PCB, PC, and PS values are saved into the system stack, and processing branches to the routine indicated by the interrupt number 10 vector. In addition, the I flag is cleared and the S flag is set. The PC value saved in the stack is the address at which the undefined instruction is stored. Processing can be restored by the RETI instruction, but is of no use, however, because the same exception occurs again.

CHAPTER 4

DELAYED INTERRUPT GENERATION MODULE

This chapter explains the functions and operations of the delayed interrupt generation module.

4.1 Overview of Delayed Interrupt Generation Module

**The delayed interrupt generation module generates the interrupt for task switching.
The hardware interrupt request can be generated/cancelled by software.**

■ Overview of Delayed Interrupt Generation Module

By using the delayed interrupt generation module, a hardware interrupt request can be generated or cancelled by software.

Table 4.1-1 shows the overview of the delayed interrupt generation module.

Table 4.1-1 Overview of Delayed Interrupt Generation Module

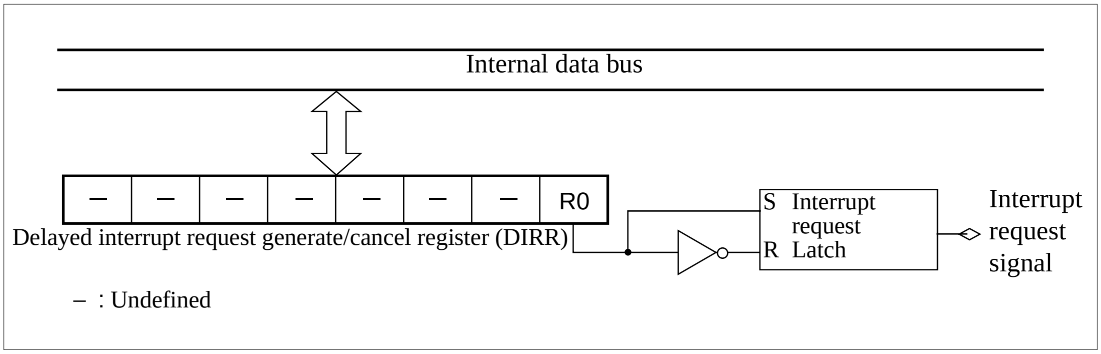
	Function and control
Interrupt factor	An interrupt request is generated by setting the R0 bit in the delayed interrupt request generate/cancel register to 1 (DIRR: R0 = 1). An interrupt request is cancelled by setting the R0 bit in the delayed interrupt request generate/cancel register to 0 (DIRR: R0 = 0).
Interrupt number	#42 ($2A_H$)
Interrupt control	An interrupt is not enabled by the DIRR register.
Interrupt flag	The interrupt flag is held in the R0 bit in the DIRR register.
EI^{2}OS	The DIRR register does not correspond to the EI^{2}OS.

4.2 Block Diagram of Delayed Interrupt Generation Module

The delayed interrupt generation module consists of the following blocks:
- **Interrupt request latch**
- **Delayed interrupt request generate/cancel register (DIRR)**

■ Block Diagram of Delayed Interrupt Generation Module

Figure 4.2-1 Block Diagram of Delayed Interrupt Generation Module



● Interrupt request latch

This latch keeps the settings (delayed interrupt request generation or cancellation) of the delayed interrupt request generate/cancel register (DIRR).

● Delayed interrupt request generate/cancel register (DIRR)

This register generates or cancels a delayed interrupt request.

■ Interrupt Number

The interrupt number used in the delayed interrupt generation module is as follows:

Interrupt number #42(2A$_H$)

4.3 Configuration of Delayed Interrupt Generation Module

This section lists registers and reset values in the delayed interrupt generation module.

■ List of Registers and Reset Values

Figure 4.3-1 List of Registers and Reset Values in Delayed Interrupt Generation Module

	bit 15	14	13	12	11	10	9	8
Delayed interrupt request generate/cancel register (DIRR) Address: 00009F$_H$	×	×	×	×	×	×	×	0

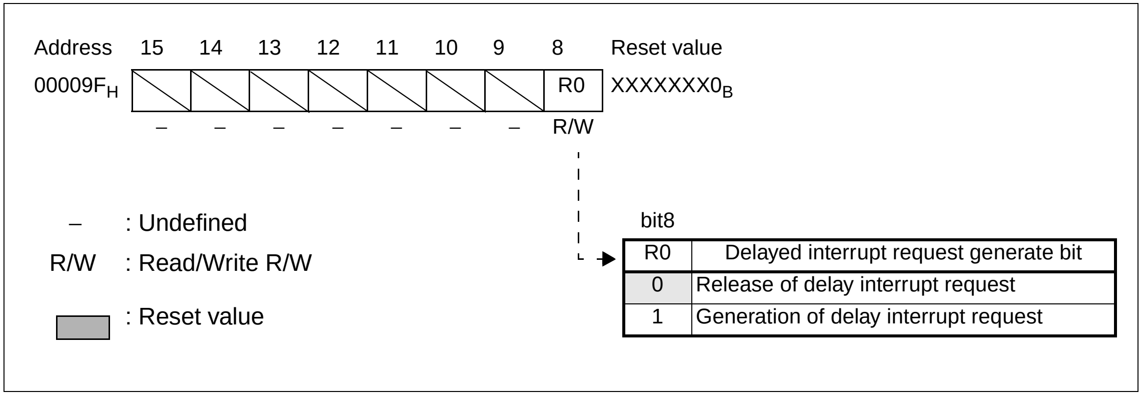
× :Undefined

4.3.1 Delayed interrupt request generate/cancel register (DIRR)

The delayed interrupt request generate/cancel register (DIRR) generates or cancels a delayed interrupt request.

■ Delayed Interrupt Request Generate/cancel Register (DIRR)

Figure 4.3-2 Delayed Interrupt Request Generate/cancel Register (DIRR)



R0	Delayed interrupt request generate bit
0	Release of delay interrupt request
1	Generation of delay interrupt request

– : Undefined

R/W : Read/Write R/W

▨ : Reset value

Table 4.3-1 Functions of Delayed Interrupt Request Generate/Cancel Register (DIRR)

Bit name		Function
bit8	R0: Delayed interrupt request generate bit	This bit generates or cancels a delayed interrupt request. **When set to "0":** Cancels delayed interrupt request **When set to "1":** Generates delayed interrupt request
bit9 to bit15	Undefined bits	**Read:** The value is undefined. **Write:** No effect

4.4 Explanation of Operation of Delayed Interrupt Generation Module

The delayed interrupt generation module has a function for generating or canceling an interrupt request by software.

■ Explanation of Operation of Delayed Interrupt Generation Module

Using the delayed interrupt generation module requires the setting shown in Figure 4.4-1 .

Figure 4.4-1 Setting for Delayed Interrupt Generation Module

	bit15	14	13	12	11	10	9	bit8
DIRR	–	–	–	–	–	–	–	R0
								❍

– : Undefined bit

❍ : Used bit

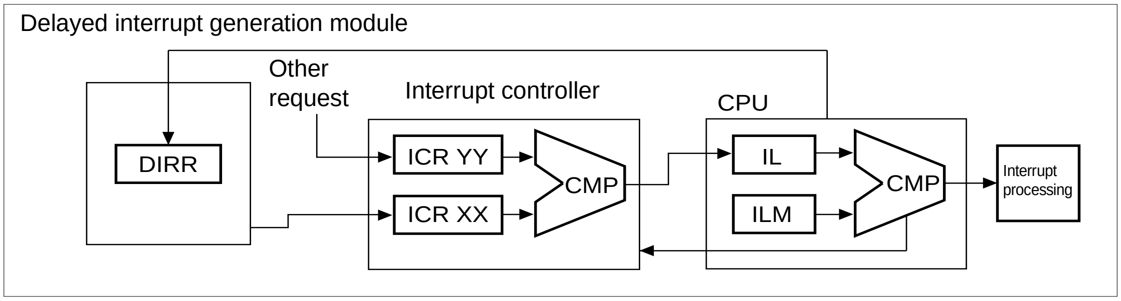
When the R0 bit in the delayed interrupt request generate/cancel register (DIRR) is set to "1" (DIRR: R0 = 1), an interrupt request is generated. There is no interrupt request enable bit.

● Operation of delayed interrupt generation module

- When the R0 bit in the delayed interrupt request generate/cancel register (DIRR) is set to "1", the interrupt request latch is set to "1" and an interrupt request is generated to the interrupt controller.
- An interrupt request is generated to the CPU when the interrupt controller prioritizes the interrupt request over other requests.
- When the interrupt level mask bit of the condition code register (CCR: ILM) is compared to the interrupt request level (ICR: IL), and the interrupt request level is higher than ILM, CPU executes the delayed interrupt processing after the instruction currently being executed is completed.
- At interrupt processing, the user program sets the R0 bit to 0 to cancel the interrupt request and performs task switching.

Figure 4.4-2 shows the operation of the delayed interrupt generation module.

Figure 4.4-2 Operation of Delayed Interrupt Generation Module



4.5 Precautions when Using Delayed Interrupt Generation Module

This section explains the precautions when using the delayed interrupt generation module.

■ Precautions when Using Delayed Interrupt Generation Module

- The interrupt processing is restarted at return from interrupt processing without setting the R0 bit in the delayed interrupt request generate/cancel register (DIRR) to "0" within the interrupt processing routine.
- Unlike software interrupts, interrupts in the delayed interrupt generation module are delayed.

4.6 Program Example of Delayed Interrupt Generation Module

This section gives a program example of the delayed interrupt generation module.

■ Program Example of Delayed Interrupt Generation Module

- Processing specification

The main program writes "1" to the R0 bit in the delayed interrupt request generate/cancel register (DIRR) to generate a delayed interrupt request and performs task switching.

- Coding example

```
ICR15   EQU    0000BFH        ;Interrupt control register
DIRR    EQU    00009FH        ;Delayed interrupt factor generate/
                               cancel register
DIRR_R0 EQU    DIRR:0         ;Delay interrupt request generating bit
;---------Main program------------------------------------
CODE    CSEG
START:                        ;Stack pointer (SP),already initialized
        AND    CCR,#0BFH      ;Interrupt disabled
        MOV    I:ICR15,#00H   ;Interrupt level 0 (strong)
        MOV    ILM,#07H       ;Setting ILM in PS to level 7
        OR     CCR,#40H       ;Interrupt enabled
        SETB   I:DIRR_R0      ;Delay interrupt request generating
        LOOP   MOV A,#00H     ;No limit loop
        MOV    A,#01H
        BRA    LOOP
;---------Interrupt program--------------------------------------
WARI:
        CLRB   I:DIRR_R0      ;Clear interrupt request flag
         :
;       User processing
;        :
        RETI                  ;Recovery from interrput
CODE    ENDS
;---------Vector setting-----------------------------------------
VECT    CSEG   ABS=0FFH
        ORG    0FF54H         ;Setting vector to interrupt #42 (2AH)
        DSL    WARI
        ORG    0FFDCH         ;Reset vector setting
        DSL    START
        DB     00H            ;Setting to single-chip mode
VECT    ENDS
        END       START
```

CHAPTER 5
CLOCKS

This chapter explains the clocks used by MB90360 series microcontrollers.

5.1 Clocks

The clock generation block controls the operation of the internal clock that controls operation of the CPU and peripheral functions. The clock generated by the clock generation block is called the machine clock. One cycle of machine clock is called one machine cycle. The clock to be supplied from a high-speed oscillator is called an oscillation clock, and the 2-frequency division of the oscillation clock is called a main clock. The 4- or 2-frequency division of the clock supplied from a low-speed oscillator or internal CR oscillation clock is called a sub-clock, and the clock by the PLL oscillation is called PLL clock.

■ Clocks

The clock generation block contains the oscillation circuit that generates the oscillation clock by connecting oscillator to oscillation pin. External clock inputted to the oscillation pins can be used as oscillation clock. The clock generation block also contains the PLL clock multiplier circuit, which generates five clocks whose frequencies are multiplication of the oscillation clock frequency. The clock generation block controls the oscillation stabilization wait interval and PLL clock multiplication as well as internal clock operation by changing the clock with a clock selector.

● Oscillation clock (HCLK)

The oscillation clock is generated either by connecting the oscillator to high-speed oscillator pins (X0,X1) or by the input of an external clock.

● Main clock (MCLK)

The main clock, whose frequency is the oscillation clock frequency divided by 2, supplies the clock input to the timebase timer and the clock selector.

● Sub-clock (SCLK)

The sub-clock is a clock by connecting the oscillator to the low-speed oscillation pins (X0A, X1A) or by inputting the external clock or the internal CR oscillation clock divided by 4 or 2. The division ratio of sub-clock is determined by SCDS bit of PLL/Subclock Control Register (PSCCR). The sub-clock can be used as operation clock of the watch timer or the low-speed machine clock.

● PLL clock (PCLK)

The PLL clock is obtained by multiplying the oscillation clock frequency with the PLL clock multiplier circuit (PLL oscillation circuit). One of five types of clocks can be selected by setting the multiplication ratio selection bits (CKSCR: CS1, CS0, PSCCR: CS2)

● Machine clock

The machine clock controls the operation of the CPU and peripheral functions. One cycle of machine clock is regarded as one machine cycle (1/ϕ). An operating machine clock can be selected from among the main clock, sub-clock, and five types of PLL clock.

Note:

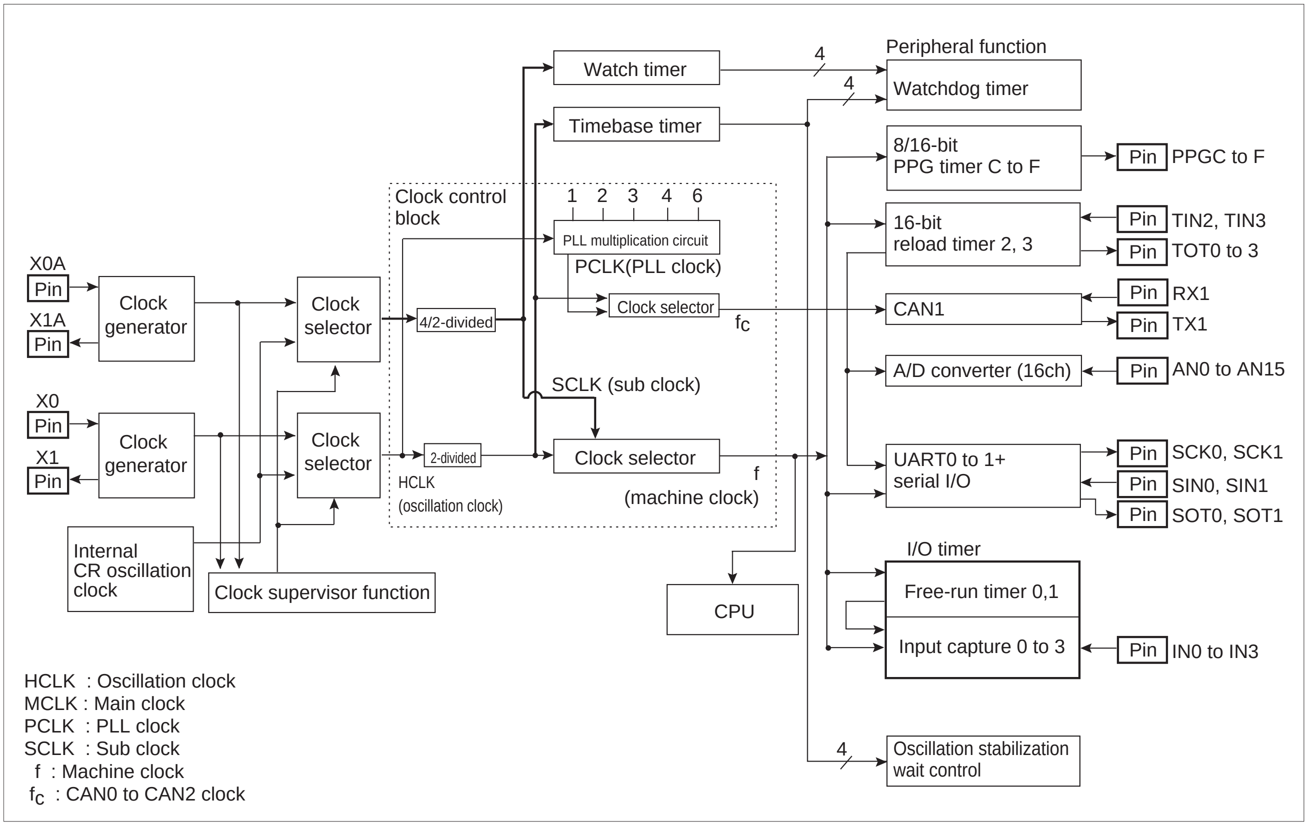
When the operating voltage is 5 V, the oscillation clock can be between 3 MHz and 16 MHz. When an external clock source is used, its frequency can be between 3 MHz and 24 MHz. The highest operating frequency for the CPU and peripheral resource is 24 MHz. However, normal operation is not guaranteed if a multiplication ratio resulting in a higher frequency than 24 MHz is specified.

Therefore, when 24 MHz external clock is inputted, 1 can be specified for the PLL clock multiplication ratio. The PLL oscillation operates at the range of 4 MHz to 24 MHz, but the PLL oscillation range differs by operation voltage and multiplication ratio. See the data sheet for the details.

■ Clock Supply Map

Since the machine clock generated in the clock generation block is supplied as the clock that controls the operation of the CPU and peripheral functions, the operation of the CPU and the peripheral functions is affected by switching between the main clock, the PLL clock and the subclock (clock mode) and by a change in the PLL clock multiplication ratio. Since some peripheral functions receive frequency-divided output from the timebase timer, a peripheral unit can select the clock best suited for this operation. Figure 5.1-1 shows the clock supply map.

Figure 5.1-1 Clock Supply Map



5.2 Block Diagram of the Clock Generation Block

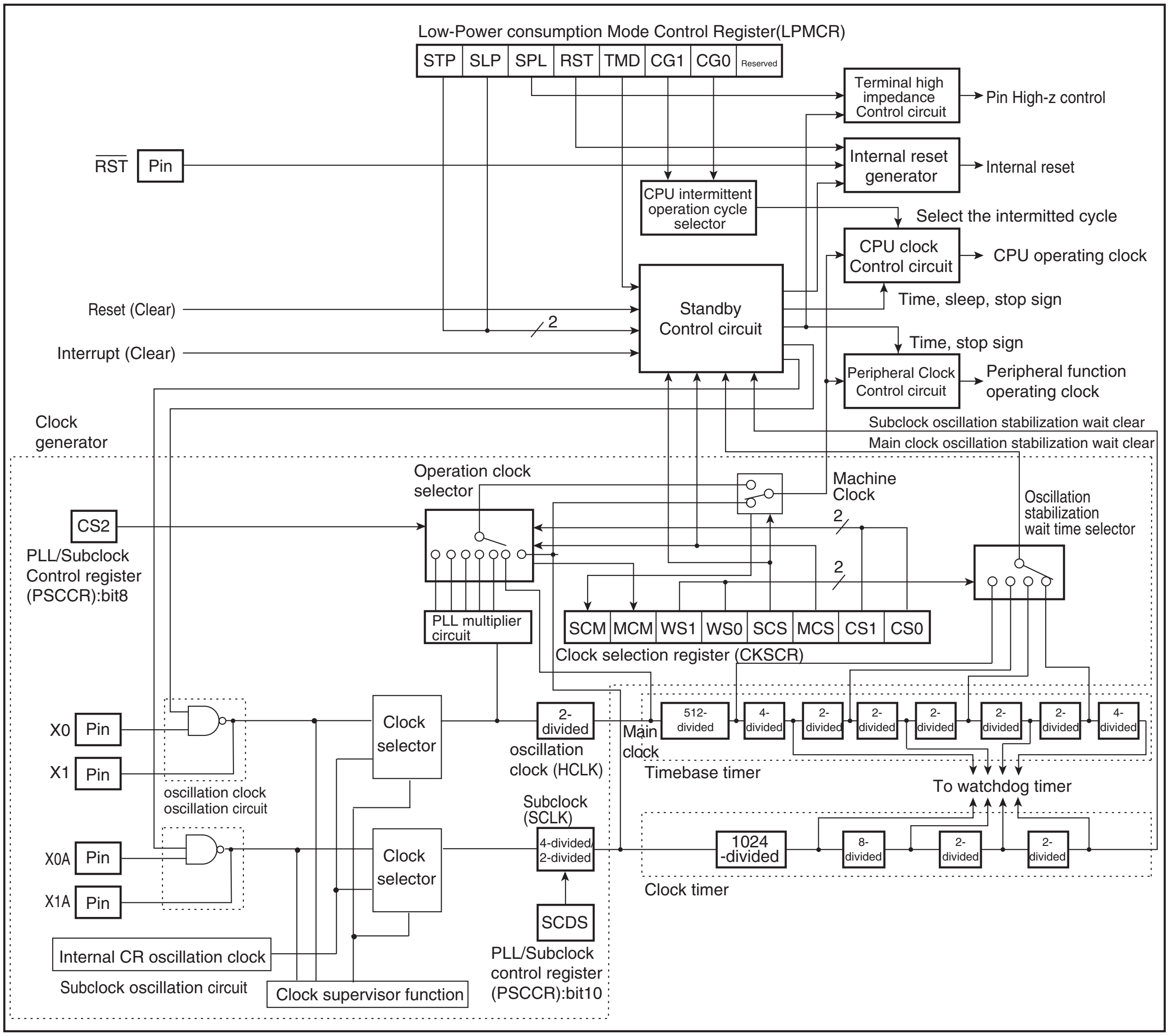
The clock generation block consists of five blocks:

- **System clock generation circuit/sub-clock generation circuit**
- **PLL multiplier circuit**
- **Clock selector**
- **Clock selection register (CKSCR)**
- **PLL/sub-clock control register (PSCCR)**
- **Oscillation stabilization wait interval selector**

■ Block Diagram of the Clock Generation Block

Figure 5.2-1 shows a block diagram of the clock generation block. The figure also includes the standby control circuit and timebase timer circuit.

Figure 5.2-1 Block Diagram of the Clock Generation Block



Low-Power consumption Mode Control Register(LPMCR)
STP
SLP
SPL
RST
TMD
CG1
CG0
Reserved
Terminal high impedance Control circuit
Pin High-z control
RST
Pin
Internal reset generator
Internal reset
CPU intermittent operation cycle selector
Select the intermitted cycle
CPU clock Control circuit
CPU operating clock
Time, sleep, stop sign
Reset (Clear)
2
Standby Control circuit
Time, stop sign
Interrupt (Clear)
Peripheral Clock Control circuit
Peripheral function operating clock
Subclock oscillation stabilization wait clear
Clock generator
Main clock oscillation stabilization wait clear
Operation clock selector
Machine Clock
2
Oscillation stabilization wait time selector
CS2
PLL/Subclock Control register (PSCCR):bit8
2
PLL multiplier circuit
SCM
MCM
WS1
WS0
SCS
MCS
CS1
CS0
Clock selection register (CKSCR)
X0
Pin
X1
Pin
Clock selector
2-divided
oscillation clock (HCLK)
Main clock
512-divided
4-divided
2-divided
2-divided
2-divided
2-divided
2-divided
4-divided
Timebase timer
oscillation clock oscillation circuit
To watchdog timer
Subclock (SCLK)
X0A
Pin
Clock selector
4-divided/2-divided
1024-divided
8-divided
2-divided
2-divided
X1A
Pin
Clock timer
SCDS
Internal CR oscillation clock
PLL/Subclock control register (PSCCR):bit10
Subclock oscillation circuit
Clock supervisor function

● Oscillation clock generation circuit

This circuit generates an oscillation clock (HCLK) by connecting an oscillator or inputting an external clock to the high-speed oscillation pins.

● Sub-clock generation circuit

This circuit generates a sub clock (SCLK) by connecting an oscillator or inputting an external clock to the low-speed oscillation pins (X0A, X1A).

● PLL multiplier circuit

This circuit multiplies the oscillation clock and supplies it as a PLL clock (PCLK) to the clock selector.

● Clock selector

From among the main clock, five different PLL clocks and subclock, the clock selector selects the clock that is supplied to the CPU and peripheral function.

● Clock selection register (CKSCR)

The clock selection register is used to switch between the oscillation clock and PLL clock and between the main clock and sub-clock, also used to select an oscillation stabilization wait interval and a PLL clock multiplier.

● PLL/Subclock Control Register (PSCCR)

The PLL/subclock control register is used to select multiplication ratio of the PLL (CS2 bit in this register in addition to CS1 and CS0 bits in the CKSCR register) and to specify division ratio (1/4 or 1/2) of the subclock.

● Oscillation stabilization wait interval selector

This oscillation stabilization wait interval selector selects an oscillation stabilization wait interval for the oscillation clock. Selection is made from among four different timebase timer outputs.

5.2.1 Register of Clock Generation Block

This section explains the register of the clock generation block.

■ Clock Selection Register and List of Reset Value

Figure 5.2-2 Clock Selection Register and List of Reset Value

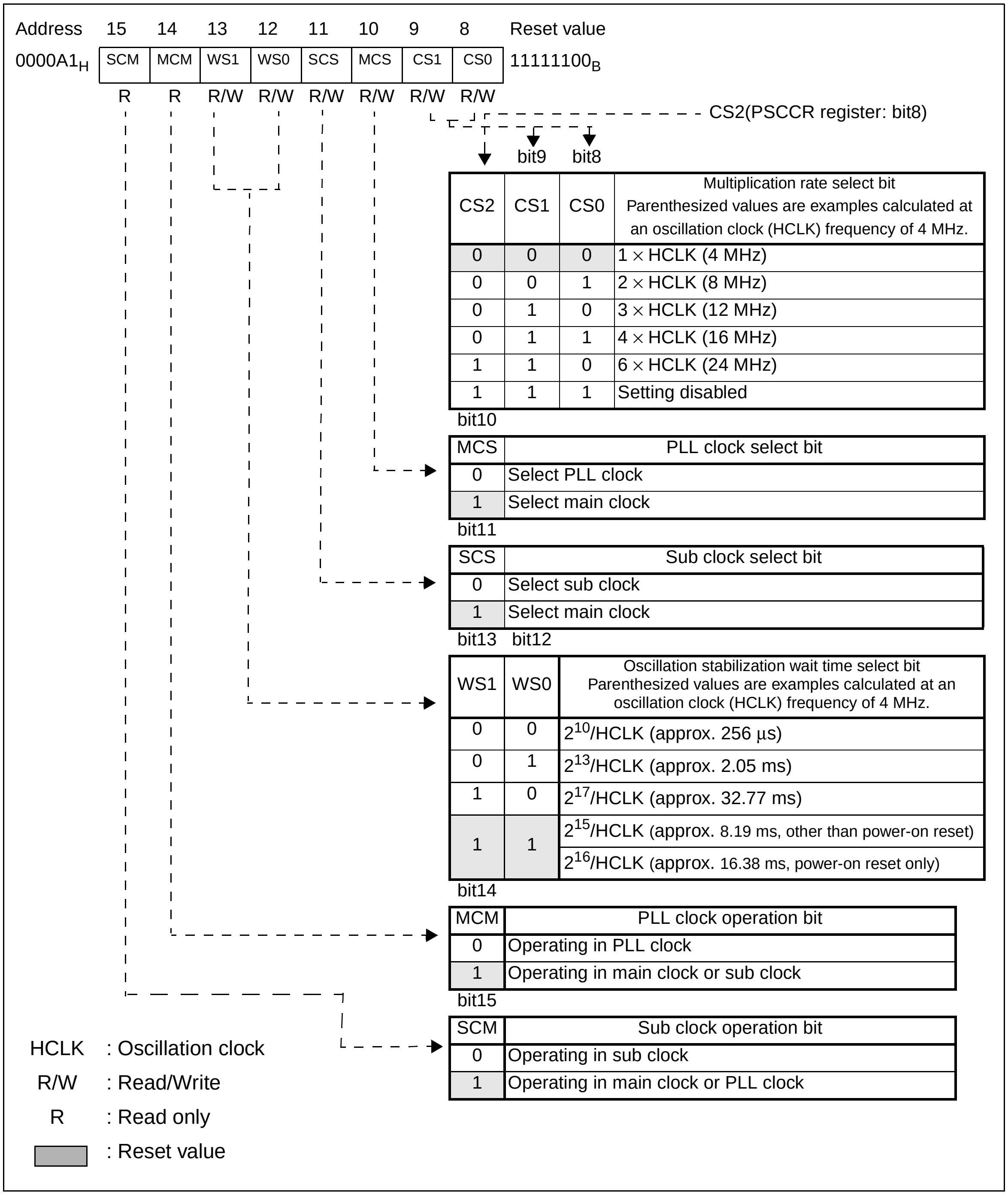
	15	14	13	12	11	10	9	8
Clock selection register (CKSCR)	1	1	1	1	1	1	0	0
PLL/subclock control register (PSCCR)	-	-	-	-	0	0	0	0

5.3 Clock Selection Register (CKSCR)

The clock selection register (CKSCR) is used to switch among the main clock, PLL clocks and subclock, also used to select an oscillation stabilization wait interval and a PLL clock multiplier.

■ Configuration of the Clock Selection Register (CKSCR)

Figure 5.3-1 Configuration of the Clock Selection Register (CKSCR)



Address	15	14	13	12	11	10	9	8	Reset value
$0000A1_H$	SCM	MCM	WS1	WS0	SCS	MCS	CS1	CS0	11111100_B
	R	R	R/W	R/W	R/W	R/W	R/W	R/W	

CS2	CS1	CS0	Multiplication rate select bit Parenthesized values are examples calculated at an oscillation clock (HCLK) frequency of 4 MHz.
0	0	0	1 × HCLK (4 MHz)
0	0	1	2 × HCLK (8 MHz)
0	1	0	3 × HCLK (12 MHz)
0	1	1	4 × HCLK (16 MHz)
1	1	0	6 × HCLK (24 MHz)
1	1	1	Setting disabled

bit10

MCS	PLL clock select bit
0	Select PLL clock
1	Select main clock

bit11

SCS	Sub clock select bit
0	Select sub clock
1	Select main clock

bit13 bit12

WS1	WS0	Oscillation stabilization wait time select bit Parenthesized values are examples calculated at an oscillation clock (HCLK) frequency of 4 MHz.
0	0	2^{10}/HCLK (approx. 256 μs)
0	1	2^{13}/HCLK (approx. 2.05 ms)
1	0	2^{17}/HCLK (approx. 32.77 ms)
1	1	2^{15}/HCLK (approx. 8.19 ms, other than power-on reset) 2^{16}/HCLK (approx. 16.38 ms, power-on reset only)

bit14

MCM	PLL clock operation bit
0	Operating in PLL clock
1	Operating in main clock or sub clock

bit15

SCM	Sub clock operation bit
0	Operating in sub clock
1	Operating in main clock or PLL clock

HCLK : Oscillation clock
R/W : Read/Write
R : Read only
▨ : Reset value

Table 5.3-1 Functions of Clock Selection Register (CKSCR) (1/2)

Bit name		Function
bit15	SCM: Sub clock operation flag bit	The bit indicates the main clock or subclock currently selected as the machine clock. When the sub clock operation flag bit (CKSCR: SCM) is "0" and the sub clock select bit (CKSCR: SCS) is "1", it indicates that the machine clock is currently switching from subclock to main clock. When the sub clock operation flag bit (CKSCR: SCM) is "1" and the sub clock select bit (CKSCR: SCS) is "0", it indicates that the machine clock is currently switching from main clock to subclock. (The writing operation will not be affected.)
bit14	MCM: PLL clock operation flag bit	The bit indicates the main clock or PLL clock currently selected as the machine clock. When the PLL clock operation flag bit (CKSCR: MCM) is "1" and the PLL clock select bit (CKSCR: MCS) is "0", it indicates that the oscillation stabilization wait time of the PLL clock is currently being taken. (The writing operation will not be affected.)
bit13 bit12	WS1, WS0: Oscillation stabilization wait time select bits	These bits are used to select an oscillation stabilization wait time required for the oscillation clock when the stop mode is canceled, when transition occurs from subclock mode to main clock mode, or when transition occurs from subclock mode to PLL clock mode. These bits are used to select one from four timebase timer outputs. Any reset causes the bit to return to the reset value. Note: Set the oscillation stabilization wait time to an appropriate value depending on the oscillator used. See 7.2.1 Reset Factors and Oscillation Stabilization Wait Times. The oscillation stabilization wait time taken when the clock mode is switched from main clock to PLL clock is fixed at 2^{14}/HCLK (about 4.1 ms during operation at an oscillation clock frequency of 4 MHz).When the CPU switches from subclock mode to PLL clock mode or when it returns from PLL stop mode to PLL clock mode, the oscillation stabilization wait time follows the values specified in these bits. The PLL clock requires an oscillation stabilization wait time of at least 2^{14}/HCLK. For switching from subclock mode to PLL clock mode and transiting to the PLL stop mode, therefore, set these bits to "10_B" or "11_B".
bit11	SCS: Sub clock select bit	This bit indicates the main clock or sub clock to be selected as the machine clock. When the machine clock is switched from the main clock to the subclock (CKSCR: SCS = 1 → 0), the main clock mode changes to the subclock mode of 1/SCLK (32.768 kHz oscillation clock frequency, operating at 4 division: approx. 130 µs) in synchronization with the subclock. When the machine clock is switched from the subclock to the main clock (CKSCR: SCS = 0 → 1), the clock mode changes from subclock mode to main clock mode after the main clock oscillation stabilization wait time is generated.Timebase timer is cleared automatically. Any reset causes the bit to return to the reset value. Notes: **1)** When both of the MCS and SCS bits contain 0, the SCS bit supersedes the MCS bit, thereby setting the subclock mode. **2)** If both the subclock select bit (CKSCR: SCS) and PLL clock select bit (CKSCR: MCS) contain 0, the sub clock is preferred. **3)** When switching from the main clock to subclock (CKSCR: SCS = 1 → 0), use the timebase timer interrupt enable bit (TBTC: TBIE) or interrupt level mask register (ILM: ILM2 to 0) to disable timebase timer interrupts before writing 0 to the subclock select bit. **4)** The 2^{14}/SCLK sub clock oscillation stabilization wait time (32.768 kHz oscillation clock frequency, operating at 4 division: approx. 2 s) is generated at power on or at cancellation of the stop mode.If the clock mode is switched from main clock mode to subclock mode, therefore, the oscillation stabilization wait time is generated.

Table 5.3-1 Functions of Clock Selection Register (CKSCR) (2/2)

<table>
<tr><th colspan="2">Bit name</th><th>Function</th></tr>
<tr><td>bit10</td><td>MCS:
PLL clock select bit</td><td>This bit indicates the main clock or PLL clock to be selected as the machine clock.
When the machine clock is switched from the main clock to the PLL clock (CKSCR: MCS = 1 → 0), the clock mode changes from main clock mode to PLL clock mode after the PLL clock oscillation stabilization wait time is generated.The timebase timer is cleared automatically.The oscillation stabilization wait time taken when the clock mode is switched from main clock to PLL clock is fixed at 2^{14}/HCLK (about 4.1 ms during operation at an oscillation clock frequency of 4 MHz).The oscillation stabilization wait time taken when the machine clock is switched from subclock mode to PLL clock mode follows the values specified in the oscillation stabilization wait time select bits (CKSCR: WS1, WS0).
Any reset causes the bit to return to the reset value.
Notes:
1) When both of the MCS and SCS bits contain 0, the SCS bit supersedes the MCS bit, thereby setting the subclock mode.
2) When switching from the main clock to PLL clock (CKSCR: MCS = 1 → 0), use the timebase timer interrupt enable bit (TBTC: TBIE) or interrupt level mask register (ILM: ILM2 to 0) to disable timebase timer interrupts before writing 0 to the PLL clock select bit.</td></tr>
<tr><td>bit9
bit8</td><td>CS1, CS0:
Multiplication rate select bits</td><td>These bits select the PLL clock multiplication rate with the CS2 bit in the PLL/subclock control register (PSCCR).
One of five types of PLL clock multiplication rate can be selected.
Any reset causes the bit to return to the reset value.
Setting of CS0, CS1, and CS2
<table>
<tr><th>CS2</th><th>CS1</th><th>CS0</th><th>PLL clock multiplication rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>× 1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>× 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>× 3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>× 4</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>× 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Setting disabled</td></tr>
</table>
Note: Setting CS2 to CS0 bits to "111_B" is prohibited.
When PSCCR: CS2 is set to "1", do not set CKSCR: CS1 and CS0 to "11_B".
When the PLL clock is selected (CKSCR: MCS = 0), writing is inhibited. To change the multiplier, write 1 to the PLL clock select bit (CKSCR: MCS), update the multiplication rate select bits (CKSCR: CS1, CS0), then set the PLL clock select bit (CKSCR: MCS) back to 0.</td></tr>
</table>

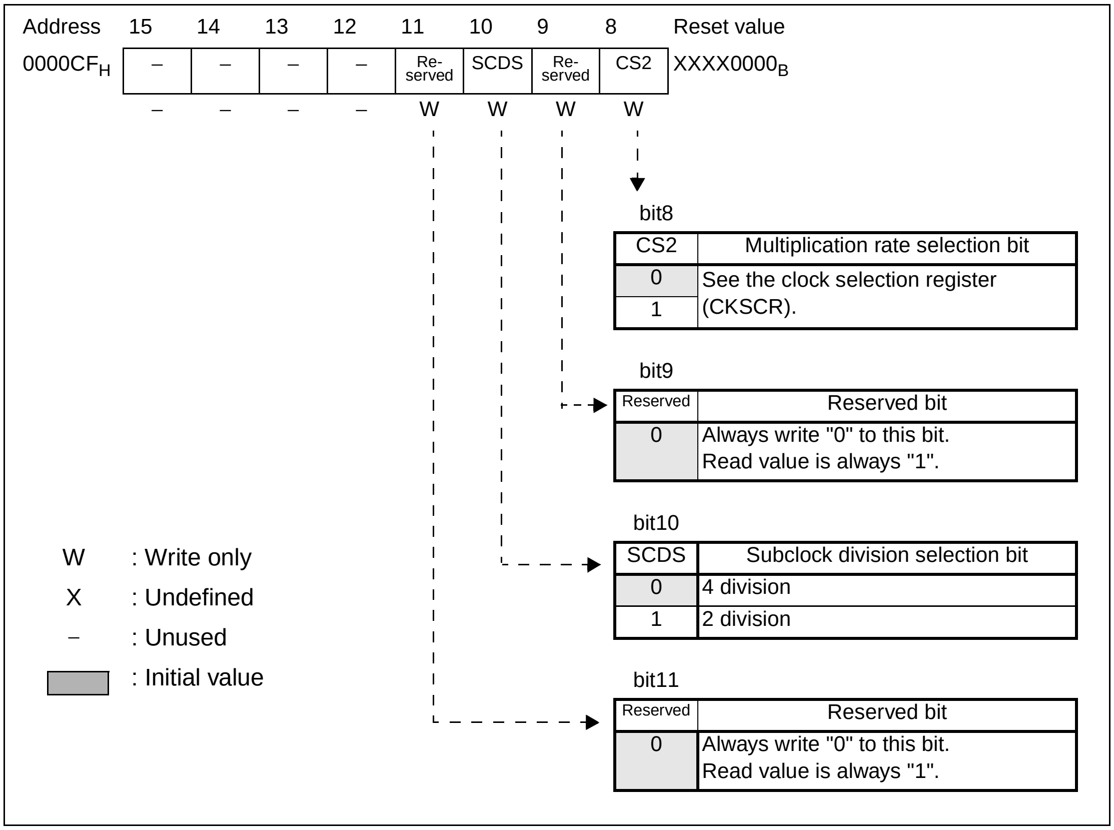
5.4 PLL/Subclock Control Register (PSCCR)

PLL/Subclock control register selects the PLL multiplication rate and subclock division rate. This register is write only. Read value of all bits is set to "1".

■ Configuration of the PLL/Subclock Control Register (PSCCR)

Figure 5.4-1 shows the configuration of the PLL/Subclock control register (PSCCR). Table 5.4-1 shows the function of each bit in the PLL/subclock control register (PSCCR).

Figure 5.4-1 Configuration of the PLL/Subclock Control Register (PSCCR)



Address	15	14	13	12	11	10	9	8	Reset value
0000CF$_H$	–	–	–	–	Re-served	SCDS	Re-served	CS2	XXXX0000$_B$
	–	–	–	–	W	W	W	W	

bit8

CS2	Multiplication rate selection bit
0	See the clock selection register (CKSCR).
1	

bit9

Reserved	Reserved bit
0	Always write "0" to this bit. Read value is always "1".

bit10

SCDS	Subclock division selection bit
0	4 division
1	2 division

bit11

Reserved	Reserved bit
0	Always write "0" to this bit. Read value is always "1".

W	: Write only
X	: Undefined
–	: Unused
▨	: Initial value

Table 5.4-1 Functional Description of Each Bit in the PLL/subclock Control Register (PSCCR)

<table>
<tr><th colspan="2">Bit name</th><th>Function</th></tr>
<tr><td>bit15
to
bit12</td><td>Unused</td><td>These bits are not used.
Writing to these bits has no effect to operation.
Read value is always "1".</td></tr>
<tr><td>bit11</td><td>Reserved bit</td><td>Always write "0" to this bit.
Read value is always "1".</td></tr>
<tr><td>bit10</td><td>SCDS:
Subclock division selection bit</td><td>The division ratio of the subclock is selected.
When "0" is written to this bit, 4 division is selected.
When "1" is written to this bit, 2 division is selected.
Read value is always "1".
This bit is initialized to "0" by all reset causes.</td></tr>
<tr><td>bit9</td><td>Reserved bit</td><td>Always write "0" to this bit.
Read value is always "1".</td></tr>
<tr><td>bit8</td><td>CS2:
Multiplication rate selection bit</td><td>This bit and CS1 and CS0 bits of the clock selection register (CKSCR) determine the PLL multiplication rate.
<table>
<tr><th>CS2</th><th>CS1</th><th>CS0</th><th>PLL clock multiplication rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>× 1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>× 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>× 3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>× 4</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>× 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Setting disabled</td></tr>
</table>
Read value is always "1".
This bit is initialized to "0" by all reset causes.
Note: When MCS or MCM bit is "0", setting CS2 to CS0 to "111_B" is prohibited.
When CKSCR: CS1 and CS0 is set to "11_B", do not set "1" to this bit.</td></tr>
</table>

Note: PSCCR register is write-only register. Read value is different from writing value. Do not use the RMW instruction (SETB/CLRB instruction).

5.5 Clock Mode

Three clock modes are provided: main clock mode, PLL clock mode and sub-clock mode.

■ Clock Mode

● Main clock mode

In main clock mode, a clock with 2-frequency division of the clock generated by connecting on oscillator or by inputting from external to the high-speed oscillation pins (X0, X1) is used.

● Sub-clock mode

In sub-clock mode, a clock with 4/2-frequency division of the clock generated by connecting an oscillator or inputting from external, or the internal CR oscillation clock to the low-speed oscillation pins (X0A, X1A) is used.

The subclock division ratio is determined by SCDS bit of PLL/subclock control register (PSCCR).

● PLL clock mode

In PLL clock mode, a PLL clock is used as the operating clock for the CPU and peripheral resources. A PLL clock multiplier is selected with the clock selection register (CKSCR: CS1 and CS0) and PLL/ subclock control register (PSCCR: CS2).

■ Clock Mode Transition

Transition among main clock mode, PLL clock mode, and sub-clock mode is performed by writing to the MCS and SCS bits of the clock selection register (CKSCR).

● Transition from main clock mode to PLL clock mode

When the MCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in main clock mode, switching from the main clock to a PLL clock occurs after the PLL clock oscillation stabilization wait interval (2^{14}/HCLK).

● Transition from PLL clock mode to main clock mode

When the MCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in PLL clock mode, switching from the PLL clock to the main clock occurs when the edges of the PLL clock and the main clock coincide (after 1 to 12 PLL clocks).

● Transition from main clock mode to sub-clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in main clock mode, switching from the main clock to a sub-clock occurs by synchronizing with the subclock.

● Transition from sub-clock mode to main clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in sub-clock mode, switching from the sub-clock to the main clock occurs after the main clock oscillation stabilization wait interval.

● Transition from PLL clock mode to sub-clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “1” to “0” in PLL clock mode, switching from the PLL clock to the sub-clock occurs.

● Transition from sub-clock mode to PLL clock mode

When the SCS bit of the clock selection register (CKSCR) is rewritten from “0” to “1” in sub-clock mode, switching from the sub-clock to a PLL clock occurs after the main clock oscillation stabilization wait interval.

■ Selection of a PLL Clock Multiplier

Writing the value from “000_B” to “011_B” and “110_B” to the CS1 and CS0 bits of the clock selection register (CKSCR) and CS2 bit of the PLL/subclock control register (PSCCR) can select five types (1 to 4 multiplication and 6 multiplication) of PLL clock multiplier.

■ Machine Clock

PLL clock, main clock, and sub-clock outputted from the PLL multiplier circuit are used as machine clock. This machine clock is supplied to the CPU and peripheral functions. The main clock, PLL clock, or sub-clock can be selected by writing to the MCS or SCS bit of the clock selection register (CKSCR).

Notes:

Even though the MCS and SCS bits of the clock selection register (CKSCR) are rewritten, machine clock switching does not occur immediately. When operating a resource that depends on the machine clock, confirm that machine clock switching has been performed by referring to the MCM and SCM bits of the clock selection register (CKSCR) before operating the resource.

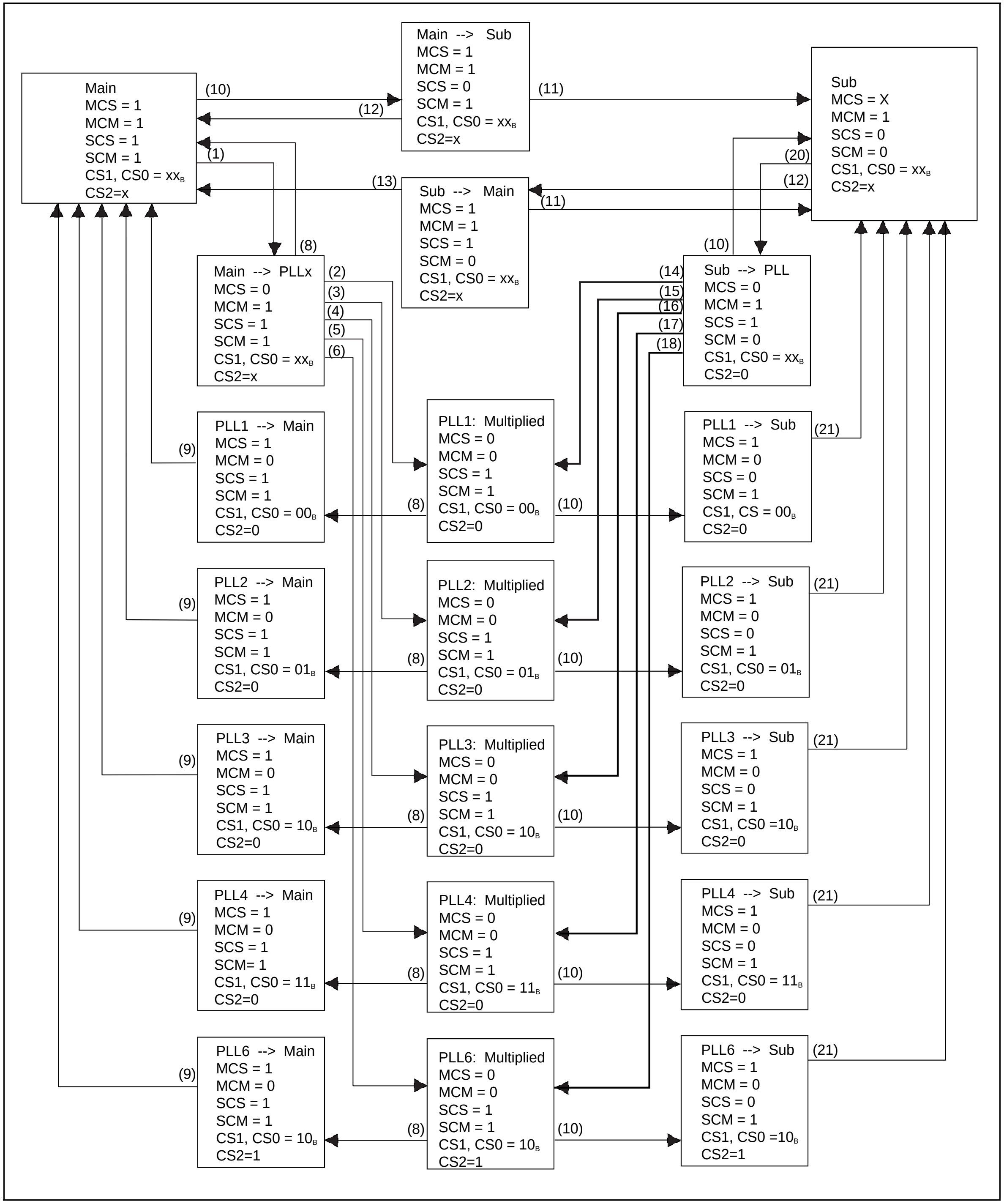
When the MCS bit of the clock selection register (CKSCR) is "0" (PLL clock mode) and when the SCS bit of the clock selection register (CKSCR) is "0" (sub-clock mode), the SCS bit is prioritized, and a transition to the sub-clock mode is occurred.

When the clock mode is switched, do not switch to other clock mode and low-power consumption mode before this switching is completed. Confirm the completion of clock mode switching by referring to the MCM and SCM bits of the clock selection register (CKSCR).

If switching to other clock mode and low-power consumption mode is performed before a transition is completed, the mode may not be switched.

Figure 5.5-1 shows the status change caused by machine clock switching.

Figure 5.5-1 Status Change Diagram for Machine Clock Selection



(1) Write "0" to MCS bit

(2) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 00_B& CS2= 0

(3) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 01_B& CS2= 0

(4) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 0

(5) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 11_B& CS2= 0

(6) Termination of PLL clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 1

(7) Write "1" to MCS bit (include reset)

(8) Synchronous timing of PLL clock and main clock

(9) Write "0" to SCS bit

(10) Synchronous timing of main clock and sub-clock

(11) Write "1" to SCS bit (MCS1)

(12) Termination of main clock oscillation stabilization wait time

(13) Termination of main clock oscillation stabilization wait time & CS1, CS0= 00_B& CS2= 0

(14) Termination of main clock oscillation stabilization wait time & CS1, CS0= 01_B& CS2= 0

(15) Termination of main clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 0

(16) Termination of main clock oscillation stabilization wait time & CS1, CS0= 11_B& CS2= 0

(17) Termination of main clock oscillation stabilization wait time & CS1, CS0= 10_B& CS2= 1

(18) Write "1" to SCS bit (MCS0)

(19) Synchronous timing of PLL clock and sub-clock

MCS : Machine clock select bit of clock selection register (CKSCR)

MCM : Machine clock display bit of clock selection register (CKSCR)

SCS : Machine clock display bit (sub) of clock selection register (CKSCR)

SCM : Machine clock select bit (sub) of clock selection register (CKSCR)

CS1, CS0 : Machine clock of clock selection register (CKSCR)

CS2 : Multiplication rate selection bit of PLL/ subclock control register (PSCCR)

Notes:

- The initial value for the machine clock setting is main clock (CKSCR: MCS = 1, SCS = 1).
- If both the SCS and MCS bits are “0”, the SCS bit takes precedence, that is, the sub-clock is selected.
- When sub-clock mode is switched to PLL clock mode, set the WS1 and WS0 bits of CKSCR to “10_B” or “11_B.”

5.6 Oscillation Stabilization Wait Interval

When the power is turned on during the oscillation clock is stopped or when stop mode is released, a time until the oscillation clock stabilizes (oscillation stabilization wait time is required immediately after oscillation starts. Also, the oscillation stabilization wait time is required when the clock mode is switched from main clock to PLL clock, main clock to sub-clock, sub-clock to main clock, and sub-clock to PLL clock.

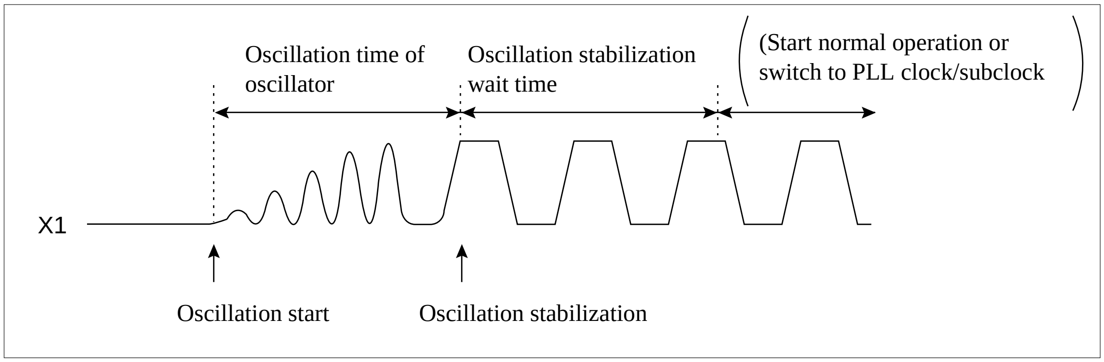
■ Oscillation Stabilization Wait Interval

Ceramic and crystal oscillators generally require several to dozens of ms to stabilize at their natural frequency (oscillation frequency) when oscillation starts. For this reason, CPU operation is not allowed immediately after oscillation starts but is allowed only after full oscillation stabilization. After the oscillation stabilization wait interval has elapsed, the machine clock is supplied to the CPU. Because the oscillation stabilization wait time depends on the type of oscillator (crystal, ceramic, etc.), the proper oscillation stabilization wait interval for the oscillator used must be selected. An oscillation stabilization wait interval is selected by setting the clock selection register (CKSCR).

When clock mode is switched from main clock to PLL clock, main clock to subclock, subclock to main clock, or subclock to PLL clock, the CPU runs in the clock mode set before switching for the oscillation stabilization wait time. After the oscillation stabilization wait time has elapsed, the CPU changes to the specified clock mode.

Figure 5.6-1 shows the operation immediately after oscillation starts.

Figure 5.6-1 Operation Immediately after Oscillation Stabilization Wait Time


Oscillation time of oscillator
Oscillation stabilization wait time
(Start normal operation or switch to PLL clock/subclock)
X1
Oscillation start
Oscillation stabilization

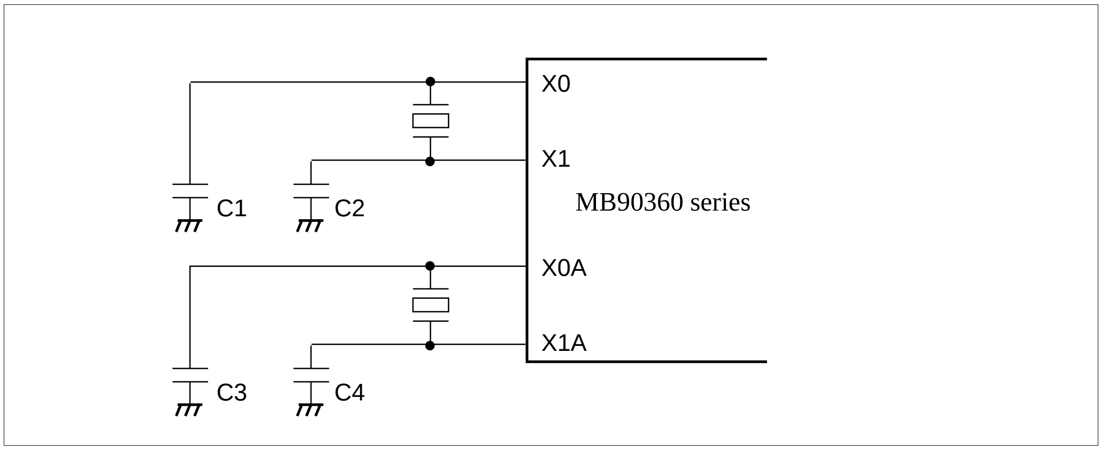
5.7 Connection of an Oscillator or an External Clock to the Microcontroller

The MB90360 series microcontroller contains a system clock generation circuit. Connecting an external oscillator to this circuit generates the system clock. Alternatively, an externally generated clock can be input to the microcontroller.

■ Connection of an Oscillator or an External Clock to the Microcontroller

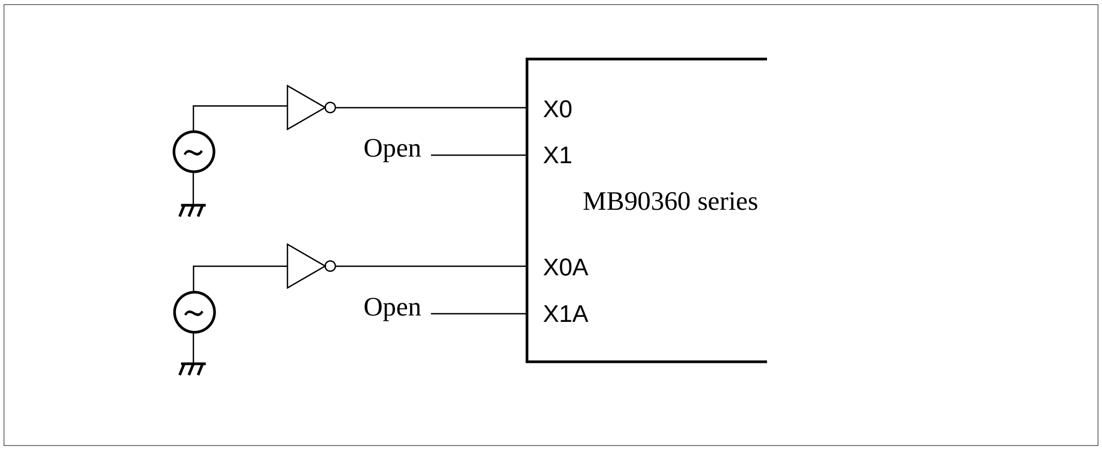
- Example of connecting a crystal or ceramic oscillator to the microcontroller

Figure 5.7-1 Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller



- Example of connecting an external clock to the microcontroller

Figure 5.7-2 Example of Connecting an External Clock to the Microcontroller



CHAPTER 6

CLOCK SUPERVISOR

This chapter explains the function and the operation of the clock supervisor. Only the product with built-in clock supervisor of the MB90360 series is valid to this function.

6.1 Overview of Clock Supervisor

The clock supervisor checks the oscillation of the main clock or a sub-clock (without "S" suffix product). When the main clock or a sub-clock stops due to some breakdowns, the control circuit of the clock supervisor switches the clock source to built-in CR oscillation clock, sets the detection flag, and generates reset.

■ Overview of Clock Supervisor

The clock supervisor checks the oscillation of the main clock or the sub-clock. If the using (main or sub) clock stops during the fixed time (20 µs to 80 µs: When the main clock is used, 160 µs to 640 µs: When the sub clock is used), the corresponding clock stop detection flag is set and reset is generated after switching the stopped clock to the CR oscillation clock.

The reset factor can be checked by the reset factor bit of watchdog timer control register (WDTC).

Supervising a main and a sub-clock can be set to the disable (watching prohibition) respectively independently.

When a sub-clock stops while the device is operating in the main clock mode, internal reset is not generated at once. When changing to the sub-clock mode, internal reset is generated. It is also possible to control the internal reset generation by the setting in this case.

When the device changes to the stop mode, the main-/sub-clock supervisor is automatically disabled (watching prohibition). Either of main or sub clock supervisor that is the condition that was enable before the stop mode changed automatically returns from disabled to enabled when returning from the stop mode.

Built-in CR oscillation clock can be used as a sub-clock of the device if the product is the external single-clock product (with "S" suffix product).

Note:

At power-on, the clock supervisor starts monitoring immediately after a lapse of the oscillation stability waiting time for the main clock.

6.2 Block Diagram of Clock Supervisor

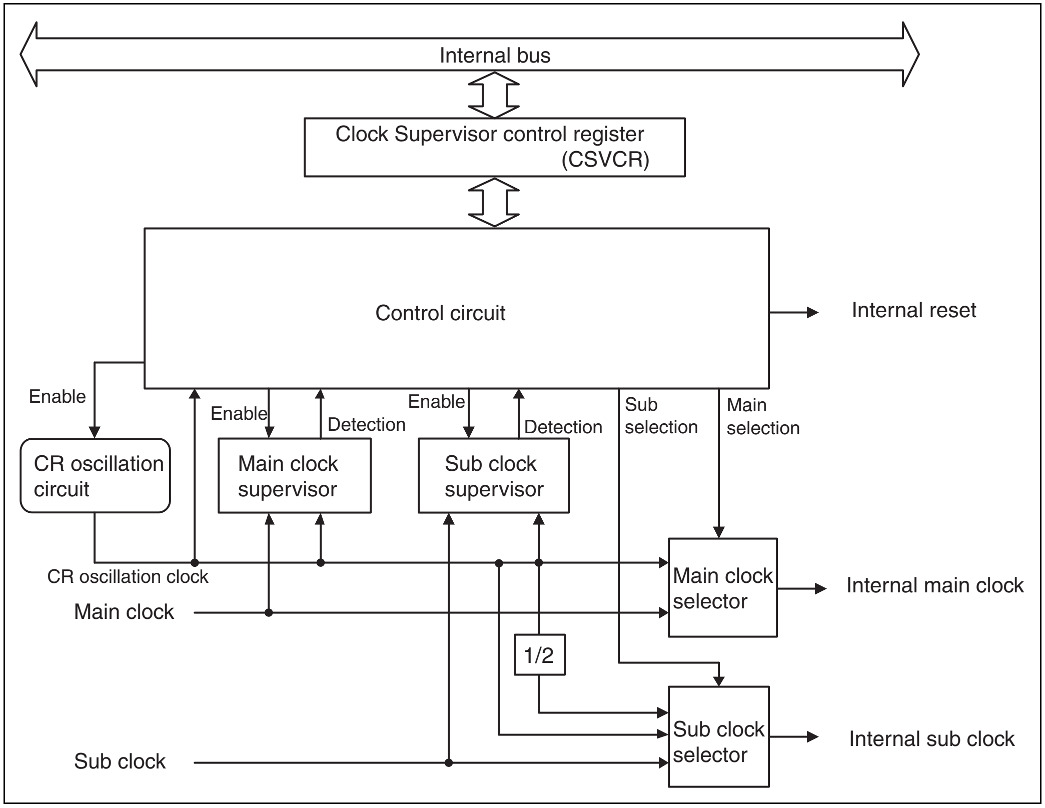
The clock Supervisor is composed of the following block:

- **Main clock supervisor**
- **Sub clock supervisor**
- **Control circuit**
- **Clock supervisor control register (CSVCR)**
- **Main clock selector**
- **Sub clock selector**
- **CR oscillation circuit**

■ Block Diagram of Clock Supervisor

Figure 6.2-1 shows the block diagram of clock supervisor.

Figure 6.2-1 Block Diagram of Clock Supervisor



- Main clock supervisor

  The oscillation of the main oscillation clock (HCLK) is supervised by using the clock from the CR oscillation circuit as a clock source.

- Sub clock supervisor

  The oscillation of the sub oscillation clock (SCLK) is supervised by using the clock from the CR oscillation circuit as a clock source.

- Control circuit

  Disable or enable for main/sub clock supervisor, existence of internal reset generation when clock halt condition is detected, and switching to CR oscillation clock of clock to be monitored are controlled by setting of clock supervisor control register (CKSCR).

- Clock supervisor control register (CKSCR)

  Disable or enable for main/sub clock supervisor, existence of internal reset generation when clock halt condition is detected, or switching to CR oscillation clock of clock to be monitored are selected.

- Main clock selector

  CR oscillation clock is outputted as main clock when the main oscillation clock is missing.

- Sub clock selector

  The divided clock of CR oscillation clock is outputted as sub clock when the sub oscillation clock is missing.

- CR oscillation circuit

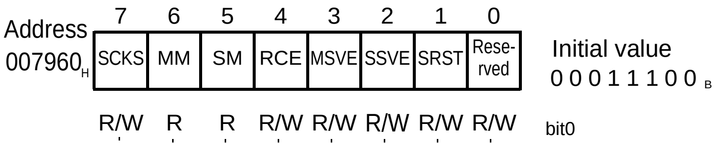
  Internal CR oscillation clock circuit. Disable/enable the CR oscillation can be selected by the control circuit.

6.3 Clock Supervisor Control Register (CSVCR)

This register switches main clock/sub clock/PLL clock, and selects the oscillation stabilization wait time and PLL clock multiplication rate.

■ Clock Supervisor Control Register (CSVCR)

Figure 6.3-1 Clock Supervisor Control Register (CSVCR)



bit0

Reserved	Reserved bit
0	Be sure to write "0" to this bit. Read value is always "0".

bit1

SRST	Sub-clock mode reset
0	No generating reset on subclock mode transition
1	Generating reset on subclock mode transition

bit2

SSVE	Sub clock supervisor enable
0	Sub clock supervisor is disabled.
1	Sub clock supervisor is enabled.

bit3

MSVE	Main clock supervisor enable
0	Main clock supervisor is disabled.
1	Main clock supervisor is enabled.

bit4

RCE	CR oscillation clock enable
0	CR oscillation clock is stopped.
1	CR oscillation clock is enabled.

bit5

SM	Sub clock missing
0	Missing sub-clock has not been detected.
1	Missing sub-clock has been detected.

bit6

MM	Main clock missing
0	Missing main clock has not been detected.
1	Missing main clock has been detected.

bit7

SCKS	Sub clock select (for "S" suffix product)
0	Not use the CR oscillation clock as sub clock
1	Use the CR oscillation clock as sub clock

R/W	:	Read/Write
R	:	Read only
▨	:	Reset value

Bit name		Function
bit7	SCKS Sub clock select	This bit permits built-in CR oscillation clock to be used as a sub-clock. Only "S" suffix product is valid to this function. "1": It is possible to change to the sub clock mode with built-in CR oscillation clock. "0": It is not possible to change to the sub clock mode. This bit is initialized to "0" by power-on reset, external reset, or low voltage detection reset in "T" suffix product. It is initialized to "0" by power-on reset or external reset without "T" suffix product. It is necessary to set this bit to "1" in the main clock mode before the sub-clock supervisor after the above-mentioned is initialized operates automatically. Moreover, after setting to "1", this bit cannot be reset to "0" by software.
bit6	MM Main clock missing	This bit indicates the oscillation missing of main clock terminal X0 and X1 was detected. "1": Missing main clock has been detected. "0": Missing main clock has not been detected. This bit is initialized to "0" by power-on reset, external reset, or low voltage detection reset with "T" suffix product. It is initialized to "0" by power-on reset or external reset without "T" suffix product.
bit5	SM Sub clock missing	This bit indicates the oscillation missing of sub clock terminal X0A and X1A was detected. No "S" suffix product is valid to this function. "1": Missing sub-clock has been detected. "0": Missing sub-clock has not been detected. This bit is initialized to "0" by power-on reset, external reset, or low voltage detection reset with "T" suffix product. It is initialized to "0" by power-on reset or external reset without "T" suffix product.
bit4	RCE CR oscillation clock enable	This bit permits built-in CR oscillation. "1": Built-in CR oscillation is enabled. "0": Built-in CR oscillation is disabled. This bit is initialized to "1" by power-on reset, external reset, or low voltage detection reset with "T" suffix product. It is initialized to "1" by power-on reset or external reset without "T" suffix product. Please set this bit in "1" after confirming both the following: - The supervisor of main and sub-clock is disabled. - MM and SM bits are "0".
bit3	MSVE Main clock supervisor enable	This bit permits monitoring for main clock oscillation. "1": Main clock supervisor is enabled. "0": Main clock supervisor is disabled. This bit is initialized to "1" by power-on reset or low voltage detection reset with "T" suffix product. It is initialized to "1" by power-on reset or external reset without "T" suffix product.
bit2	SSVE Sub clock supervisor enable	This bit permits monitoring for sub clock oscillation. "1": Sub clock supervisor is enabled. "0": Sub clock supervisor is disabled. This bit is initialized to "1" by power-on reset or low voltage detection reset with "T" suffix product. It is initialized to "1" by power-on reset or external reset without "T" suffix product.
bit1	SRST Sub-clock mode reset	This bit permits the reset output when transmitting from main clock/PLL clock mode to the sub-mode with sub-clock breakdown. "1": Output reset. "0": Not output reset This bit is initialized to "0" by power-on reset, external reset, or low voltage detection reset with "T" suffix product. It is initialized to "0" by power-on reset or external reset without "T" suffix product.
bit0	Reserved bit	This bit is reserved. Be sure to write "0" to this bit. Read value is always "0".

6.4 Operating Mode of Clock Supervisor

This section explains all the operating modes of the Clock Supervisor.

■ Operating Mode in Initialized State

The CR oscillation circuit, the main clock supervisor and the sub-clock supervisor are enabled before the clock supervisor control register (CSVCR) is set by the user program.

- After power-on reset or reset of the low voltage detection, the CR oscillation circuit is enabled with "T" suffix product. After power-on reset or external reset, the CR oscillation circuit is enabled without "T" suffix product.
- If the main clock goes off after a lapse of the oscillation stability waiting time (2^{11}/HCLK), the main clock monitor function will be immediately enabled to cause reset to occur.
- If the main clock goes off before a lapse of the oscillation stability waiting time after power-on reset, the main clock monitor function will cause reset after a lapse of the 2^{12} cycle of CR oscillation clock (approximately 41 ms for the CR oscillation of 100 kHz).
- If the main clock goes off during the period of power-on reset, the device will retain the reset state.
- After it passes of 2^{18} cycles of the CR oscillation clock (For about 2.6 s:CR oscillation 100 kHz), the sub-clock supervisor is valid.
- When the main clock is stopped on the main clock supervisor enable state, the main clock is replaced with the CR oscillation clock, MM bit is set to one, and the reset is generated.
- When the sub clock is stopped on the sub clock mode, the sub clock is replaced with the CR oscillation two dividing frequency clock, SM bit is set to one, and the reset is generated. When the sub clock is stopped on the main clock mode, the sub clock is replaced with the CR oscillation two dividing frequency clock, SM bit is set to one. However, the reset is not generated at the sub-clock mode transition because the initial value of SRST bit is "0".

■ Prohibition Setting of CR Oscillation Circuit and Clock Supervisor

In the following settings, it is assumptions that the CR oscillation circuit, the main clock supervisor, and the sub-clock supervisor are operating.

- MSVE(CSVCR:bit3) is set to 0 and the main clock supervisor is set disable.
- SSVE(CSVCR:bit2) is set to 0 and the sub clock supervisor is set disable.
- The RCE bit (bit4 of CSVCR) is set to 0 and the CR oscillation circuit is set disable. Please set it after checking that the main clock and the sub-clock supervisor are disabled, and both SM and MM (bit4 of CSVCR) are 0. Do not set RCE to 0 when either SM or MM is one.

■ Reoperating Setting of CR Oscillation Circuit and Clock Supervisor

In the following settings, it is assumptions that the CR oscillation circuit, the main clock supervisor, and the sub-clock supervisor are stopped.

- RCE(CSVCR:bit4) is set to 1 and the CR oscillation circuit is set enable.
- MSVE(CSVCR:bit3) is set to 1 and the main clock supervisor is set enable. Please note the programming of software to do after 10 µs or more has passed since the CR oscillation circuit was set enable.

- The sub-clock supervisor is operated by setting SSVE(CSVCR:bit2) to 1. Please note the programming of software to do after 10 µs or more has passed since the CR oscillation circuit was set enable.

■ Sub-clock Mode

The main clock supervisor automatically becomes disable at the sub-clock mode. The content of enable bit MSVE never changes. If the main clock was lost after oscillation stability waiting time of 2^{11}/HCLK (about 0.51 ms: at external 4 MHz) or before the oscillation stability waiting time ends, the main clock supervisor is valid after it passes of 2^{12} cycles of the CR oscillation clock (about 41 ms: at CR oscillation 100 kHz) at transition from main clock mode to sub clock mode.

■ Sub-clock Mode Transition Operating When Sub-clock Has Already Stopped

The behavior that shifts to the sub-clock mode depends on the state of the SRST bit when the stop of the sub-clock is detected by the sub-clock supervisor while the device is operating in the main clock mode.

- When SRST is set to 0 (initial state), the reset is not generated at transition to the sub-clock mode. In this case, the CR oscillation clock is used as a sub-clock at transition to the sub-clock mode.
- When SRST is set to 1, the reset is generated at transition to the sub-clock mode.

■ Stop Mode

CR oscillation circuit, the main clock, and the sub-clock supervisor automatically become disable at transition to the stop mode, when all of these functions are enable. Each enable bit of the clock supervisor control register is not changed. Therefore, after it is released from the stop mode, each enable/disable state of CR oscillation circuit and clock supervisor keep the state before they changes to the stop mode.

- The CR oscillation circuit immediately becomes enable after released from the stop mode.
- If the main clock was lost after oscillation stability waiting time of 2^{11}/HCLK (about 0.51 ms: at external 4 MHz) or before the oscillation stability waiting time ends, the main clock supervisor is enabled after it passes of 2^{12} cycles of the CR oscillation clock (about 41 ms: at CR oscillation 100 kHz).
- After it passes of 2^{18} cycles of the CR oscillation clock (For about 2.6 s:CR oscillation 100 kHz), the sub-clock supervisor is valid.

■ Sub-clock Mode with External Single Clock Product

In the sub-clock mode with external single clock product ("S" suffix product), the CR oscillation clock can be used as a sub-clock.

To use this function, SCKS (bit7 of CSVCR) is set to 1 and SRST is set to 0 (initial value). This function can not be used with the external dual clock products (no "S" suffix product).

■ Reset Check By Clock Supervisor

To check whether reset was executed by the clock supervisor, the WDTC register is read with software and the reset factor is checked. When ERSR (bit4 of WDTC) is set, the factor is a reset from an external terminal or a reset by the clock supervisor (include low voltage detection/CPU operating detection reset in "T" suffix products). If both SM and MM bits (bit5 and bit6 of CSVCR) are 0, the reset factor is an external reset (include low voltage detection/CPU operating detection reset in "T" suffix products). If SM is 1, the reset factor is a sub-clock lost. If MM is 1, the reset factor is a main-clock lost.

CHAPTER 7
RESETS

This chapter describes resets for the MB90360-series microcontrollers.

7.1 Resets

If a reset is generated, the CPU immediately stops the current execution process and waits for the reset to be cleared. The CPU then begins processing at the address indicated by the reset vector.

The four causes of a reset are as follows

- **Power-on reset**
- **External reset request via the $\overline{\text{RST}}$ pin**
- **Software reset request**
- **Watchdog timer overflow**
- **Low voltage detection reset request (product with "T"-suffix)**
- **CPU operation detection reset request (product with "T"-suffix)**
- **Clock supervisor reset request (MB90367/T(S))**

■ Causes of a Reset

Table 7.1-1 lists the causes of a reset.

Table 7.1-1 Cause of a Reset

Reset	Cause	Machine clock	Watchdog timer	Oscillation stabilization wait
Power-on	At power on	Main clock (MCLK)	Stop	Yes
External pin	L level input to $\overline{\text{RST}}$ pin	Main clock (MCLK)	Stop	None
Software	Write "0" to internal reset signal generation bit (RST) of low-power consumption mode control register (LPMCR)	Main clock (MCLK)	Stop	None
Watchdog timer	Watchdog timer overflow	Main clock (MCLK)	Stop	None
Low voltage detection reset (with "T"-suffix)	When low voltage (4.0 V ± 0.3 V) is detected	Main clock (MCLK)	Stop	None
CPU operation detection reset (with "T"-suffix)	When CPU operation detection counter overflows	Main clock (MCLK)	Stop	None
Clock supervisor reset	When failure of main clock/subclock is detected	Internal CR oscillation clock	Stop	None

MCLK: Main clock (oscillation clock frequency divided by 2)

● Power-on reset

A power-on reset is generated when the power is turned on. The oscillation stabilization wait times is fixed to 2^{16} oscillation clock cycles (2^{16}/HCLK) (approx. 16.38 ms, oscillating at 4 MHz). When the oscillation stabilization wait time has elapsed, the reset is executed.

● External reset

An external reset is generated by the L level input to an external reset pin ($\overline{\text{RST}}$ pin). The minimum required period of the L level is at least 500 ns. Reset operation is performed after oscillation stabilization wait time elapses.

Note:

If the reset cause is generated during a write operation, the CPU waits for the reset to be cleared after completion of the instruction only for reset requests via the $\overline{\text{RST}}$ pin. Therefore, the normal write operation is completed even though a reset is inputted concurrently. However, note that the following two points.

Note that a reset may prevent the data transfer requested by a string-processing instruction from being completed because the reset is accepted before a specified number of counters are transferred.

At external bus access, if the cycle is exceeded a certain period by RDY input, the reset is accepted forcibly without waiting the completion of instruction. Forcible reset is accepted within 16 machine cycles.

When returning to the main clock mode by the external reset pin ($\overline{\text{RST}}$ pin) from the stop mode, sub-clock mode, sub-sleep mode, and watch mode, input L level for at least oscillation time of oscillator* + 100 µs.

*: Oscillation time of oscillator is the time that amplitude reaches 90%. It takes several to dozens of ms for crystal oscillators, hundreds of µs to several ms for FAR/ceramic oscillators, and 0 ms for external clocks.

When returning to the main clock mode by the external reset pin ($\overline{\text{RST}}$ pin) from the timebase timer mode, input L level for at least 100 µs.

● Software reset

A software reset is generated an internal reset by writing "0" to the RST bit of the low-power consumption mode control register (LPMCR). The oscillation stabilization wait time is not required for a software reset.

● Watchdog reset

A watchdog reset is generated by a watchdog timer overflow that occurs when "0" is not written to the WTE bit of the watchdog timer control register (WDTC) within a given time after the watchdog timer is activated. The oscillation stabilization wait time is not required for watchdog reset.

● Low voltage detection reset

The low voltage detection reset is generated when the low voltage (4.0 V ± 0.3 V) is detected.

The oscillation stabilization wait time is not required for the low voltage detection reset.

● CPU operation detection reset

The CPU operation detection reset is 20-bit counter that the source oscillation is count-locked. If the CL bit of the low voltage/CPU operation detection reset is not cleared within a specified time after activation, the reset is generated.

The oscillation stabilization wait time is not required for the CPU operation detection reset.

● Clock supervisor reset

When the failure of the main clock/subclock is detected, the clock supervisor reset is generated.

The oscillation stabilization wait time is not required for the clock supervisor reset.

Definition of clocks

HCLK: Oscillation clock frequency

MCLK: Main clock frequency

φ: Machine clock (CPU operating clock) frequency

1/φ: Machine cycle (CPU operating clock period)

See "5.1 Clocks", for details.

Note:

When the reset is occurred in the stop mode or sub-clock mode, the oscillation stabilization wait time of 2^{15}/HCLK (approx. 8.19 ms, using at HCLK = 4 MHz oscillation) is required.

See "5.6 Oscillation Stabilization Wait Interval" for details.

7.2 Reset Cause and Oscillation Stabilization Wait Times

The MB90360 series has seven reset causes. The oscillation stabilization wait time for a reset depends on the reset cause.

■ Reset Causes and oscillation Stabilization Wait Times

Table 7.2-1 summarizes reset causes and oscillation stabilization wait times.

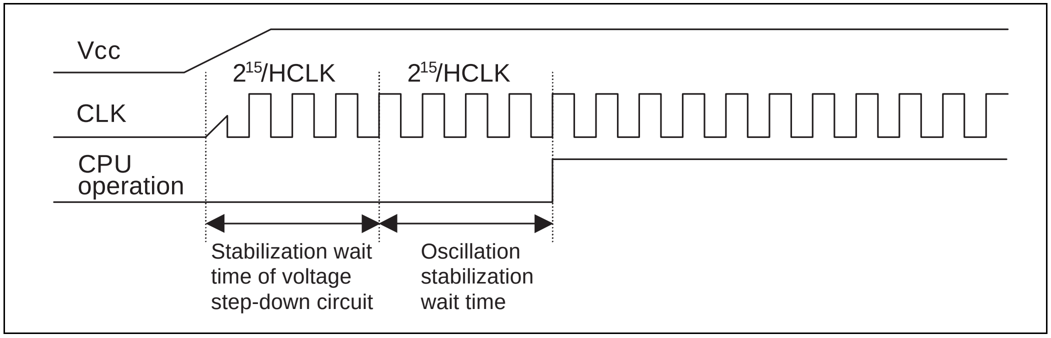
Table 7.2-1 Reset Causes and oscillation Stabilization Wait Times

Reset	Reset cause	Oscillation stabilization wait time The parenthesized values are provided when oscillation clock frequency operates at 4 MHz
Power-on	Power-on	2^{16}/HCLK (approx. 16.38 ms)
Watchdog	Watchdog timer overflow	None Note: However, the WS1 and WS0 bits are initialized to "11".
External	L input from $\overline{\text{RST}}$ pin	None Note: However, the WS1 and WS0 bits are initialized to "11".
Software	Write "0" to RST bit of low-power consumption mode control register (LPMCR)	None Note: However, the WS1 and WS0 bits are initialized to "11".
Low voltage detection *1	When low voltage is detected	None Note: However, the WS1 and WS0 bits are initialized to "11".
CPU operation detection *1	When CPU operation detection counter overflows	None Note: However, the WS1 and WS0 bits are initialized to "11".
Clock supervisor *2	When failure of main clock/subclock is detected	None Note: However, the WS1 and WS0 bits are initialized to "11".

HCLK: Oscillation clock frequency
WS1, WS0: Oscillation stabilization wait time select bit of clock selection register (CKSCR)
*1: Product with T-suffix
*2: For MB90F367/T(S), MB90367/T(S)

Figure 7.2-1 shows the oscillation stabilization wait times at a power-on reset.

Figure 7.2-1 Oscillation Stabilization Wait Times at a Power-on Reset


Vcc
2^{15}/HCLK
2^{15}/HCLK
CLK
CPU operation
Stabilization wait time of voltage step-down circuit
Oscillation stabilization wait time

Note:

Ceramic and crystal oscillators generally require an oscillation stabilization wait time of several milliseconds to some tens of milliseconds, until stabilization at a natural frequency is attained after starts oscillation. A proper oscillation stabilization wait time must be set for the particular oscillator used.

See "5.6 Oscillation Stabilization Wait Interval", for details about oscillation stabilization wait times.

■ Oscillation Stabilization Wait and Reset State

A reset operation in response to a power-on reset and other resets during stop mode or sub-clock mode is performed after the oscillation stabilization wait time has elapsed. This time interval is generated by the timebase timer. If the external reset has not been cleared after the interval, the reset operation is performed after the external reset is cleared.

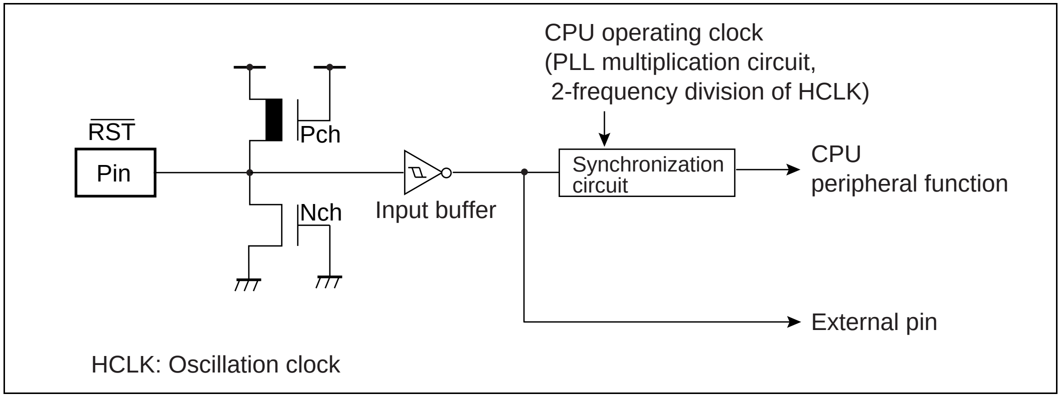
7.3 External Reset Pin

The external reset pin ($\overline{\text{RST}}$ pin) is an input pin used exclusively for a reset. Inputting an L level signal generates an internal reset. For the MB90360-series, resets are generated in synchronization with the CPU operating clock. However, initialization of external pin is asynchronous with the CPU operating clock.

■ Block Diagrams of the External Reset Pin

- Block diagram of the external reset pin

Figure 7.3-1 Block Diagram of the External Reset Pin



Note:

Inputs to the $\overline{\text{RST}}$ pin are accepted during cycles in which memory is not affected to prevent memory from being destroyed by a reset during a write operation.

A clock is required to initialize the internal circuit. In particular, an operation with an external clock requires clock input together with reset input.

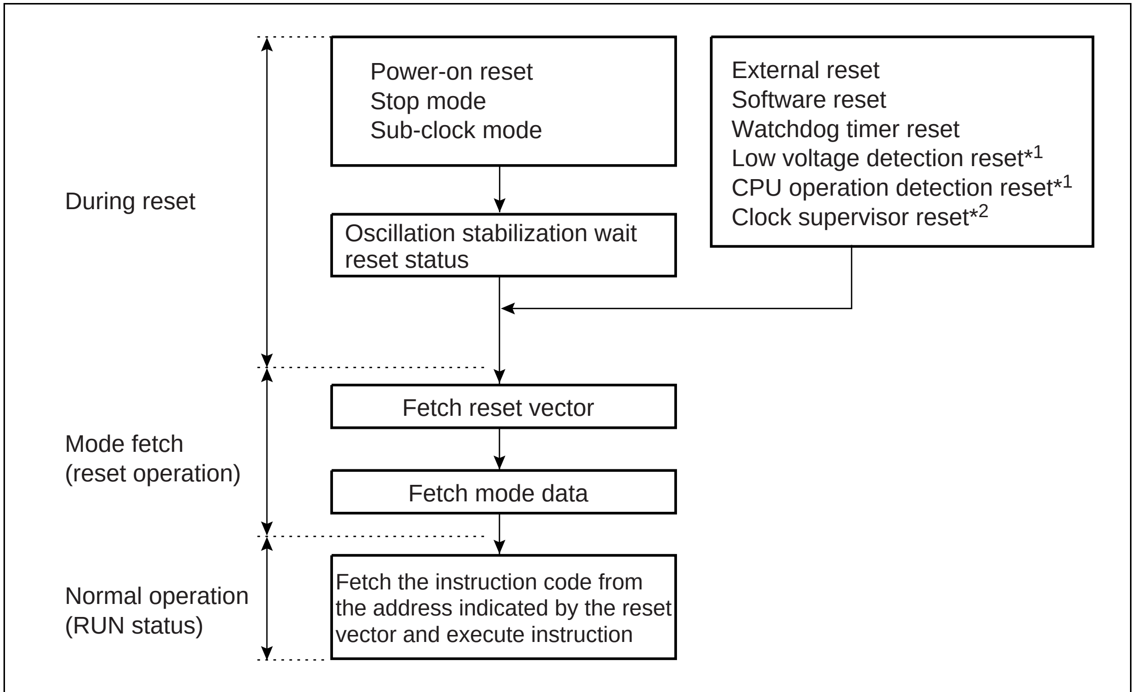
7.4 Reset Operation

When the reset signal is inactivated, the reset vector and mode data is fetched from the predetermined locations depending on the setting of the mode pins. This operation, the mode fetch, then defines the operation mode of the CPU and the start address of the first instruction. For the power on reset, reset from the stop mode or sub-clock mode, the mode fetch is performed after the oscillation stabilization wait time is elapsed.

■ Overview of Reset Operation

Figure 7.4-1 shows the reset operation flow.

Figure 7.4-1 Reset Operation Flow


During reset
Power-on reset
Stop mode
Sub-clock mode
External reset
Software reset
Watchdog timer reset
Low voltage detection reset*1
CPU operation detection reset*1
Clock supervisor reset*2
Oscillation stabilization wait reset status
Mode fetch (reset operation)
Fetch reset vector
Fetch mode data
Normal operation (RUN status)
Fetch the instruction code from the address indicated by the reset vector and execute instruction

*1: Product with T-suffix

*2: For MB90F367/T(S), MB90367/T(S)

■ Mode Pins

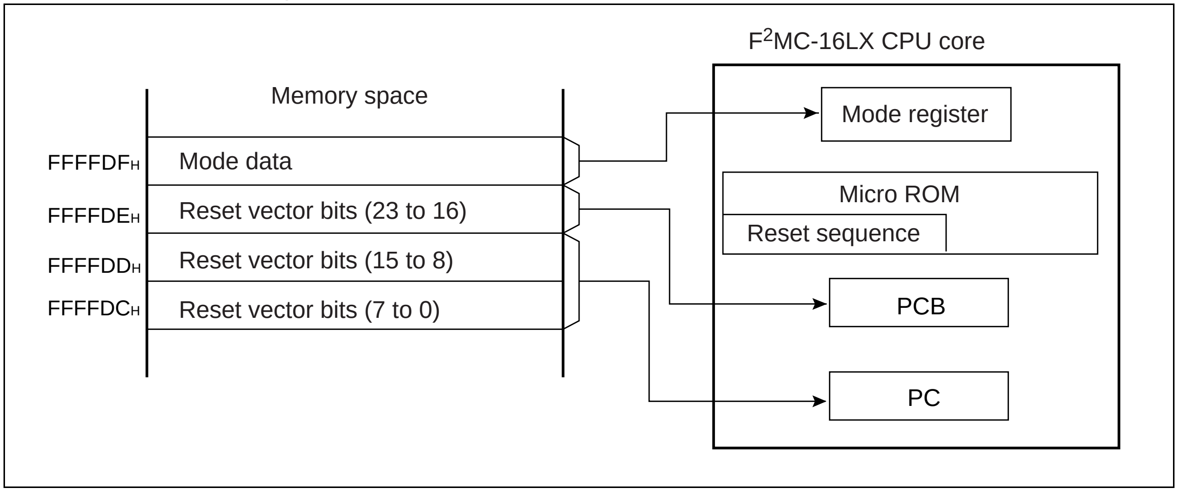
Setting the mode pins (MD0 to MD2) specifies how to fetch the reset vector and the mode data. Fetching the reset vector and the mode data is performed in the reset sequence. See "9.1.1 Mode Pins", for details on mode pins.

■ Mode Fetch

When the reset is cleared, the CPU transfers the reset vector and the mode data to the appropriate registers in the CPU core by hardware. The reset vector and mode data are allocated to the four bytes from "FFFFDC$_H$" to "FFFFDF$_H$". The CPU outputs these addresses to the bus immediately after the reset is cleared and then fetches the reset vector and mode data. Using mode fetching, the CPU can begin processing at the address indicated by the reset vector.

Figure 7.4-2 shows the transfer of the reset vector and mode data.

Figure 7.4-2 Transfer of Reset Vector and Mode Data



● Mode data (address: FFFFDF$_H$)

Only a reset operation changes the contents of the mode register. The mode register setting is valid after a reset operation. See "9.1.2 Mode Data", for details on mode data.

● Reset vector (address: FFFFDC$_H$ to FFFFDE$_H$)

The reset vector points to the start address after the reset operation. The CPU starts to execute the first instruction stored in the start address.

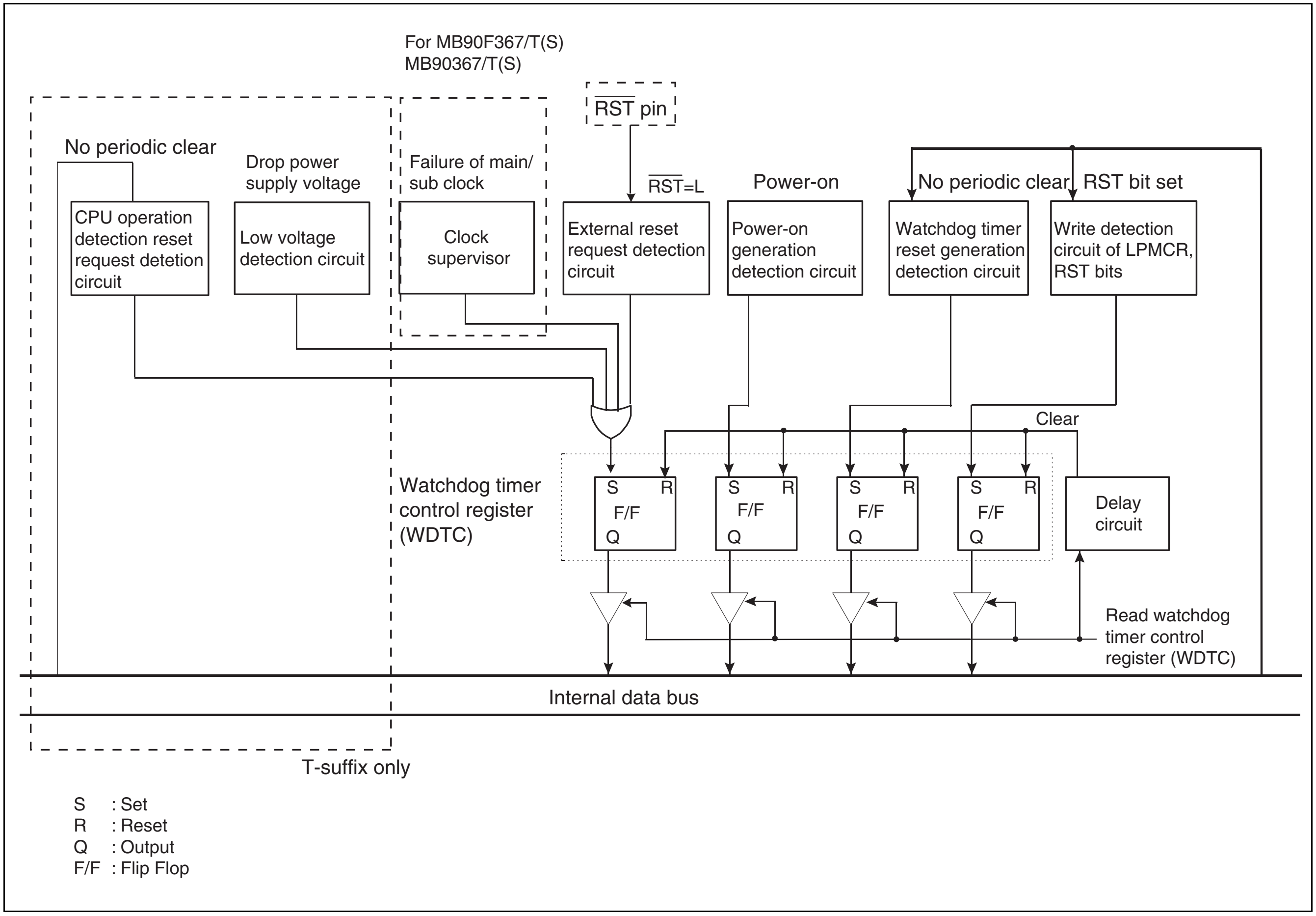
7.5 Reset Cause Bits

A reset cause can be identified by reading the watchdog timer control register (WDTC).

■ Reset Cause Bits

As shown in Figure 7.5-1 , a flip-flop is associated with each reset cause. The contents of the flip-flops are obtained by reading the watchdog timer control register (WDTC). If the cause of a reset must be identified after the reset has been cleared, the value read from the WDTC should be processed by the software and a branch made to the appropriate program.

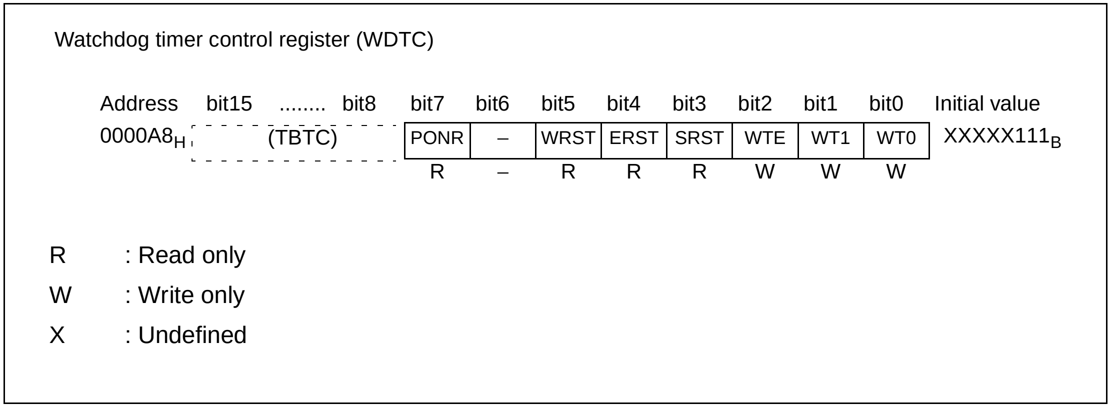
Figure 7.5-1 Block Diagram of Reset Cause Bits


For MB90F367/T(S)
MB90367/T(S)
RST pin
No periodic clear
Drop power supply voltage
Failure of main/ sub clock
RST=L
Power-on
No periodic clear
RST bit set
CPU operation detection reset request detetion circuit
Low voltage detection circuit
Clock supervisor
External reset request detection circuit
Power-on generation detection circuit
Watchdog timer reset generation detection circuit
Write detection circuit of LPMCR, RST bits
Clear
Watchdog timer control register (WDTC)
S R F/F Q
S R F/F Q
S R F/F Q
S R F/F Q
Delay circuit
Read watchdog timer control register (WDTC)
Internal data bus
T-suffix only
S : Set
R : Reset
Q : Output
F/F : Flip Flop

■ Correspondence between reset cause bits and reset causes

Figure 7.5-2 shows the configuration of the reset cause bits of the watchdog timer control register (WDTC). Table 7.5-1 maps the correspondence between the reset cause bits and reset causes. See “Watchdog timer control register (WDTC)” in "12.1 Overview of Watchdog Timer", for details.

Figure 7.5-2 Configuration of Reset Cause Bits (watchdog timer control register)



Watchdog timer control register (WDTC)

Address	bit15 .... bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
$0000A8_H$	(TBTC)	PONR	–	WRST	ERST	SRST	WTE	WT1	WT0	$XXXXX111_B$
		R	–	R	R	R	W	W	W	

R : Read only

W : Write only

X : Undefined

Table 7.5-1 Correspondence between Reset Cause Bits and Reset Causes

Reset cause	PONR	WRST	ERST	SRST
Generation of power-on reset request	1	X	X	X
Generation of reset request due to watchdog timer overflow	Δ	1	Δ	Δ
External reset request from $\overline{\text{RST}}$ pin, Low voltage detection reset (product with T-suffix)*1 CPU operation detection reset (product with T-suffix)*2 Clock supervisor reset (MB90F367/T(S), MB90367/T(S))	Δ	Δ	1	Δ
Generation of software reset request	Δ	Δ	Δ	1

Δ: Previous state retained

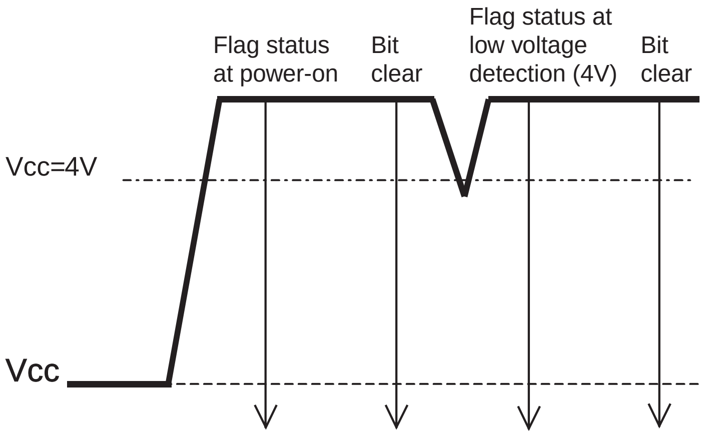
X: Undefined

*1: When the low voltage detection reset request is used, the CPUF bit of the low voltage/CPU operation detection reset control register (LVRC) is also set to "1".

*2: When the CPU operation detection reset request is used, the CPUF bit of the low voltage/CPU operation detection reset control register (LVRC) is also set to "1".

■ Status of Reset Cause Bit and Low Voltage Detection Bit

Figure 7.5-3 Status of Reset Cause Bit and Low Voltage Detection Bit



	(1)		(2)		(3)		(4)
PONR bit (power-on)	1	→	0	→	0	→	0
ERST bit (external reset input, CPU operation detection, or LVRF = 1)	1 or 0	→	0	→	1	→	0
LVRF bit* (low voltage detection 4V ± 0.3V)	1 or 0	→	0	→	1	→	0

*: The LVRF bit exist in the low voltage/CPU operation detection reset control register (LVRC).

(1) At power-on

Power-on reset bit (PONR) , ERST, and LVRF are set to "1" at power on.

(2) Bit clear

Bit is cleared by reading the WDTC register and by writing "0" to LVRF.

(3) At low voltage detection (4.0 V ± 0.3 V)

The LVRF and ERST bits are set to "1" at low voltage detection of V_{CC} = 4.0 V ± 0.3 V.

(4) Bit clear

Bit is cleared by reading the WDTC register and by writing "0" to LVRF.

■ Notes about Reset Cause Bits

● Multiple reset causes generated at the same time

When multiple reset causes are generated at the same time, the corresponding reset cause bits of the watchdog timer control register (WDTC) are also set to "1". If, for example, an external reset request via the $\overline{\text{RST}}$ pin and the watchdog timer overflow occur at the same time, the ERST and the WRST bits are both set to "1".

● Power-on reset

For a power-on reset, because the PONR bit is set to "1" but all other reset cause bits are undefined, the software should be programmed so that it will ignore all reset cause bits except the PONR bit if it is "1".

● Clearing the reset cause bits

The reset cause bits are cleared only when the watchdog timer control register (WDTC) is read. Any bit corresponding to a reset cause that has already been generated is not cleared even though another reset is generated (a setting of "1" is retained).

Note:

If the power is turned on under conditions where no power-on reset occurs, the value in WDTC register may not be guaranteed.

7.6 Status of Pins in a Reset

This section describes the status of pins when a reset occurs.

■ Status of Pins during a Reset

The status of pins during a reset depends on the settings of mode pins (MD2 to MD0).

About status of each pins during reset, please see "8.7 Status of Pins in Standby Mode and during Hold and Reset".

● When internal vector mode has been set: (MD2 to MD0 = "011$_B$")

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

■ Status of Pins after Mode Data is Read

The status of pins after mode data has been read depends on the mode data (M1 and M0).

● When single-chip mode has been selected (M1, and M0 = "00$_B$")

All I/O pins (resource pins) are high impedance, and mode data is read from the internal ROM.

Note:

For those pins that change to high impedance when a reset cause is generated, confirm that devices connected to the pins do not malfunction.

CHAPTER 8
LOW-POWER CONSUMPTION MODE

This chapter explains the low-power consumption mode of MB90360 series microcontrollers.

8.1 Overview of Low-Power Consumption Mode

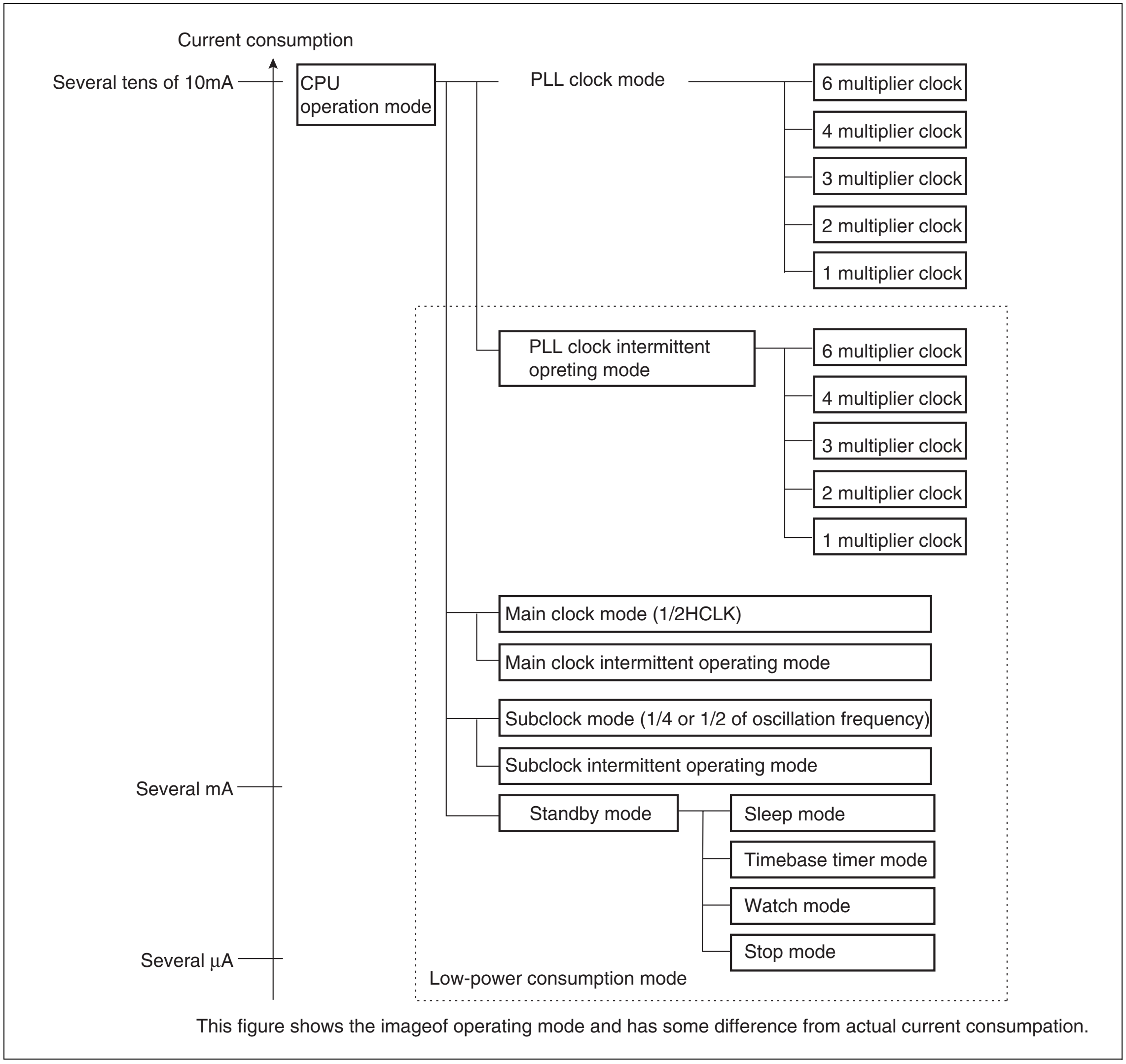
The MB90360 series has the following CPU operating modes, any of which can be used depending on operating clock selection and clock oscillation control:

- **Clock mode : main clock mode, PLL clock mode, or sub-clock mode**
- **CPU intermittent operating mode : main clock intermittent operating mode, PLL clock intermittent operating mode, or sub-clock intermittent operating mode**
- **Standby mode : sleep mode, stop mode, watch mode, or timebase timer mode**

■ CPU Operating Modes and Current Consumption

Figure 8.1-1 shows the relationship between the CPU operating modes and current consumption.

Figure 8.1-1 CPU Operating Mode and Current Consumption



This figure shows the imageof operating mode and has some difference from actual current consumpation.

■ Clock Mode

● PLL clock mode

In this mode, a PLL clock that is a multiple of the oscillation clock (HCLK) is used to operate the CPU and peripheral functions.

● Main clock mode

In this mode, the main clock, with the oscillation clock (HCLK) frequency divided by 2 is used to operate the CPU and peripheral functions. In the main clock mode, the PLL multiplier circuit is inactive.

● Sub-clock mode

In this mode, the sub-clock (SCLK) is used to operate the CPU and peripheral functions. The sub-clock can select a clock frequency divided by 2 or 4 of clock from external sub-clock pin or internal CR oscillation clock.

In the sub-clock mode, the main clock and PLL multiplier circuit are inactive.

The subclock oscillation stabilization wait time of 2^{14}/SCLK (Approx. 2 s @32.768 kHz oscillation clock frequency, 1/4 division) takes place when power-on and reactivation from stop mode. If a transition from main clock mode to subclock mode is performed during this oscillation stabilization wait time, actual transition may be delayed.

Reference:

For the clock mode, see "5.5 Clock Mode".

■ CPU Intermittent Operating Mode

In this mode, the CPU is operated intermittently while high-speed clock pluses are supplied to peripheral functions, thereby reducing power consumption. In this mode, intermittent clock pulses are supplied only to the CPU while it is accessing a register, internal memory, peripheral function, or external unit.

■ Standby Mode

In this mode, the standby control circuit stops supplying the clock to the CPU or peripheral functions or stops the oscillation clock itself (HCLK), thereby reducing power consumption.

● Sleep mode

The sleep mode stops the operation clock to the CPU during operation in each clock mode. The CPU stops, and the peripheral function operates the clock before the transition to the sleep mode. The sleep mode is divided into the main sleep mode, PLL sleep mode before the transition to sleep mode.

● Watch mode

The watch mode operates the sub-clock (SCLK), watch timer, and low voltage detection circuit only. The main clock and PLL clock stop. All peripheral functions other than the watch timer and low voltage detection circuit stop.

● Timebase timer mode

The timebase timer mode operates the oscillation clock (HCLK), sub-clock (SCLK), timebase timer, watch timer, and low voltage detection circuit only. All peripheral functions other than the timebase timer, watch timer, and low voltage detection circuit stop.

● Stop mode

The stop mode stops the oscillation clock (HCLK) and sub-clock (SCLK) during operation in each clock mode, and all functions other than low voltage detection circuit stop. Data can be retained at the lowest power consumption.

Note:

When the clock mode is switched, do not switch to other clock mode and low-power consumption mode before this switching is completed. Confirm the completion of clock mode switching by referring to the MCM and SCM bits of the clock selection register (CKSCR).

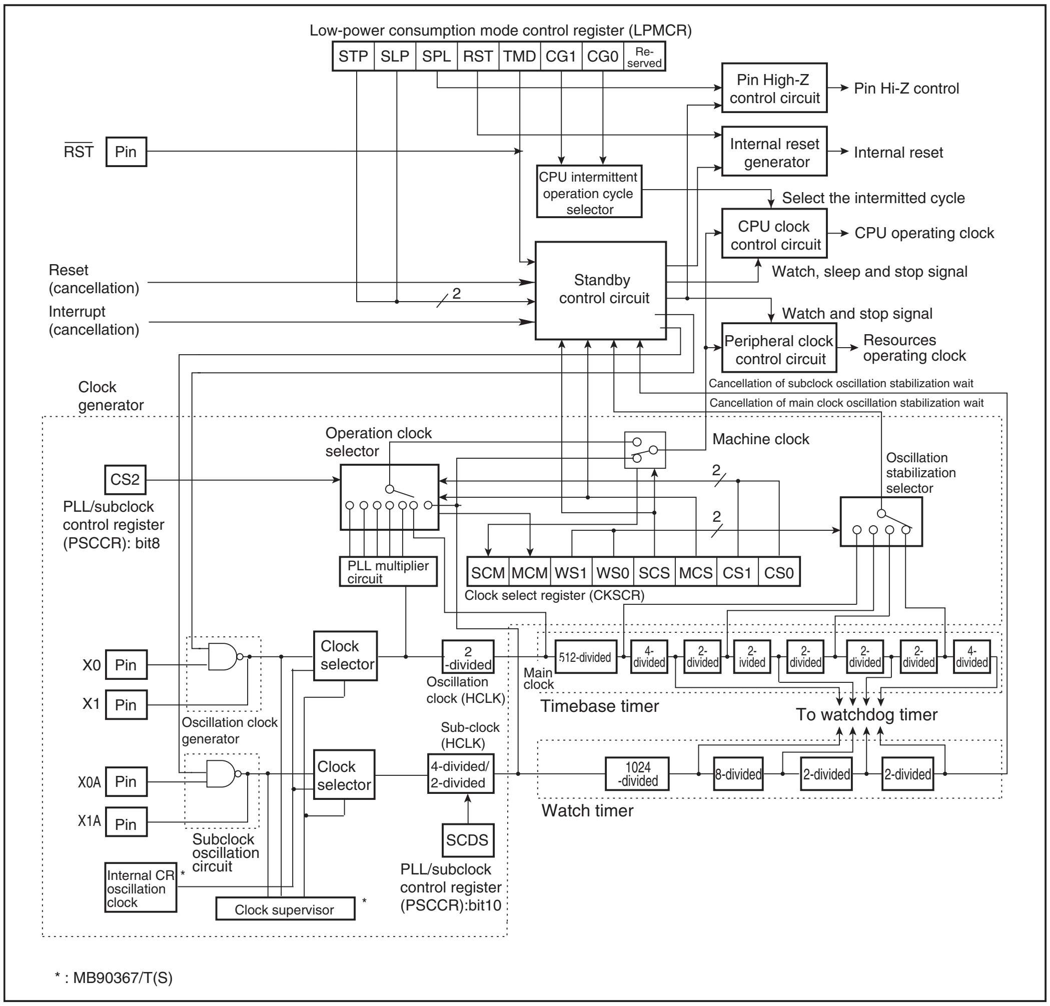
If the mode is switched to other clock mode and low-power consumption mode before completion of switching, the mode may not be switched.

8.2 Block Diagram of the Low-Power Consumption Control Circuit

This section shows the block diagram of the low-power consumption control circuit.

■ Block Diagram of the Low-Power Consumption Control Circuit

Figure 8.2-1 Block Diagram of the Low-Power Consumption Control Circuit



- ● CPU intermittent operation selector

This selector selects the halt cycle count of the CPU clock during the CPU intermittent operation mode.

- ● Standby control circuit

CPU clock control and peripheral clock control circuits switch the CPU operation clock and peripheral function operation clock to transit to and cancel the standby mode.

● CPU clock control circuit

This circuit controls clocks supplied to the CPU.

● Pin high-impedance control circuit

This circuit makes I/O pins high-impedance in the watch mode, timebase timer mode and stop mode.

● Internal reset generation circuit

This circuit generates an internal reset signal.

● Low-power consumption mode control register (LPMCR)

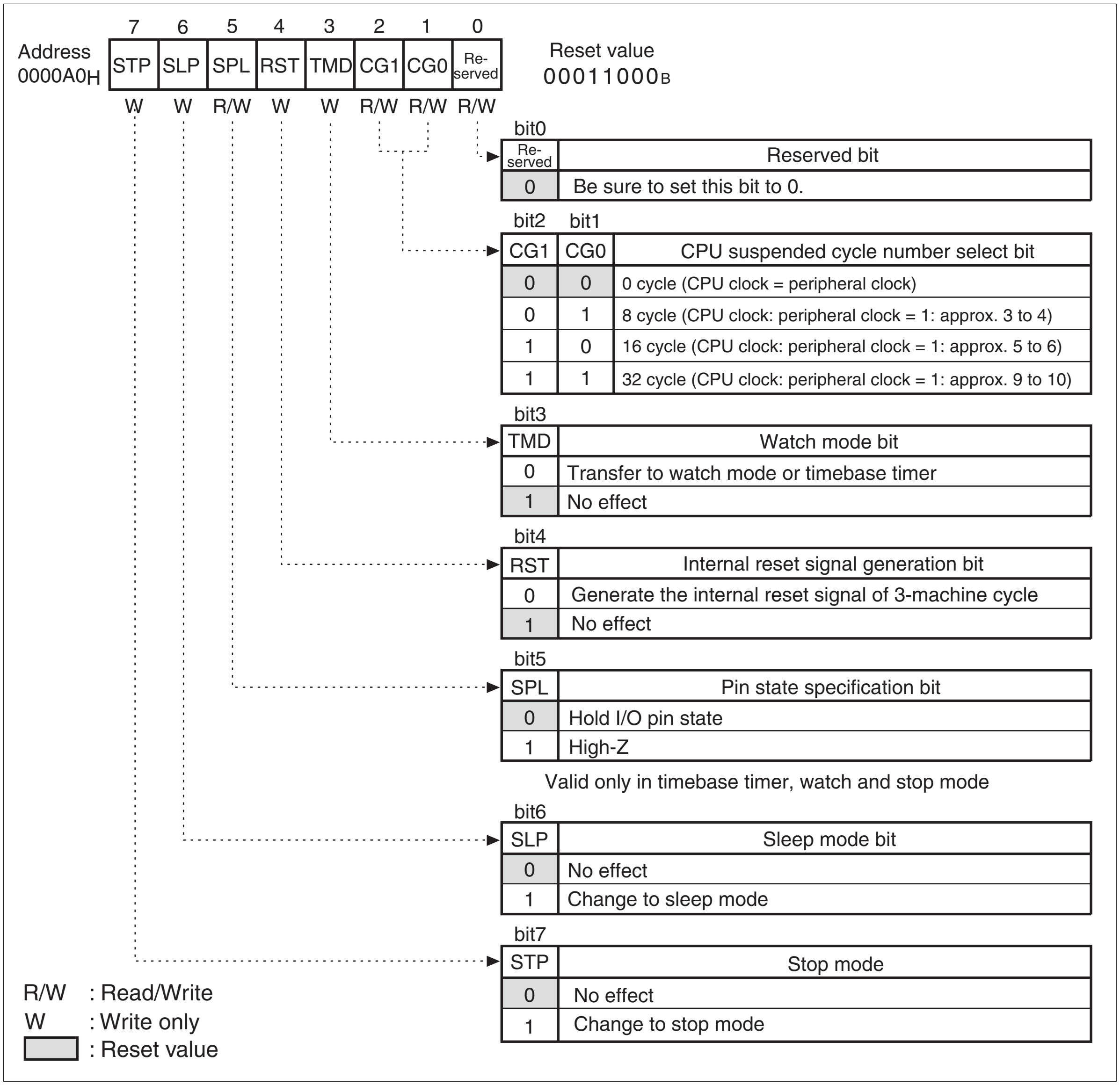
This register is used to switch to and release the standby mode and to set the CPU intermittent operation mode.

8.3 Low-Power Consumption Mode Control Register (LPMCR)

This register switches to or releases the low-power consumption mode. This register also generates the internal reset signal and sets the halt cycle count during the CPU intermittent operation mode.

■ Low-Power Consumption Mode Control Register (LPMCR)

Figure 8.3-1 Configuration of the Low-power Consumption Mode Control Register (LPMCR)



	7	6	5	4	3	2	1	0
Address 0000A0H	STP	SLP	SPL	RST	TMD	CG1	CG0	Re-served
	W	W	R/W	W	W	R/W	R/W	R/W

Reset value
00011000B

bit0

Re-served	Reserved bit
0	Be sure to set this bit to 0.

bit2 CG1	bit1 CG0	CPU suspended cycle number select bit
0	0	0 cycle (CPU clock = peripheral clock)
0	1	8 cycle (CPU clock: peripheral clock = 1: approx. 3 to 4)
1	0	16 cycle (CPU clock: peripheral clock = 1: approx. 5 to 6)
1	1	32 cycle (CPU clock: peripheral clock = 1: approx. 9 to 10)

bit3

TMD	Watch mode bit
0	Transfer to watch mode or timebase timer
1	No effect

bit4

RST	Internal reset signal generation bit
0	Generate the internal reset signal of 3-machine cycle
1	No effect

bit5

SPL	Pin state specification bit
0	Hold I/O pin state
1	High-Z

Valid only in timebase timer, watch and stop mode

bit6

SLP	Sleep mode bit
0	No effect
1	Change to sleep mode

bit7

STP	Stop mode
0	No effect
1	Change to stop mode

R/W : Read/Write
W : Write only
▨ : Reset value

Table 8.3-1 Functions of Low-power Consumption Mode Control Register (LPMCR)

Bit name		Function
bit7	STP: Stop mode bit	This bit transits to the stop mode. **When the bit is set to "0":** No effect. **When the bit is set to "1":** The CPU enters the stop mode. **Read:** "0" is always read. • The bit is initialized to "0" when a reset or external interrupt occurs.
bit6	SLP: Sleep mode bit	This bit shift to sleep mode **When the bit is set to "0":** No effect. **When the bit is set to "1":** The CPU enters the sleep mode. **Read:** "0" is always read. • The bit is initialized to "0" when a reset or external interrupt occurs. • When the STP and SLP bits are set to "1" at the same time, the STP bit supersedes the SLP bit, causing a transition to stop mode.
bit5	SPL: Pin state specification bit	The bit is used to set the state of input/output pins after transition to the stop mode, watch mode, or timebase timer mode. **When the bit is set to "0":** The current level of input/output pins is held. **When the bit is set to "1":** The I/O pins enter a high impedance state. • The bit is initialized to "0" at a reset.
bit4	RST: Internal reset signal generation bit	This bit generates software reset. **When the bit is set to "0":** An internal reset signal for three machine cycles is generated. **When the bit is set to "1":** No effect **Read:** "1" is always read.
bit3	TMD: Watch mode bit	This bit shift to watch mode or timebase timer mode **When the bit is set to "0":** If the main clock mode or PLL clock mode is used, the bit transits to the timebase timer mode. If the sub-clock mode is used, the bit transits to the watch mode. **When the bit is set to "1":** No effect • The bit is set to "1" when a reset or interrupt occurs. **Read:** "1" is always read.
bit1 bit2	CG1, CG0: CPU suspended cycle number select bits	These bits are used to set the halt cycle count of the CPU clock in the CPU intermittent operation mode. • Any reset causes the bit to return to the reset value.
bit0	Reserved: reserved bit	Always set this bit to "0".

Notes:

- Switching to a low-power consumption mode is performed by writing the low-power consumption mode control register (LPMCR). Only the instructions listed in Table 8.3-2 should be used for this purpose. If other instructions are used for switching to a low-power consumption mode, operation cannot be assured.
- The standby mode transition instruction in Table 8.3-2 must always be followed by an array of instructions highlighted by a line below.

```
MOV  LPMCR, #H’ xx    ; The low-power consumption mode transition instruction in Table 8.3-2
NOP
NOP
JMP  $+3              ; jump to next instruction
MOV  A, #H’ 10        ; any instruction
```

  The devices does not guarantee its operation after returning from the standby mode if you place an array of instructions other than the one enclosed in the dotted line.
- To access the low-power consumption mode control register (LPMCR) with C language, refer to "■ Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode" in "8.8 Usage Notes on Low-Power Consumption Mode".
- When word-length is used for writing the low-power consumption mode control register (LPMCR), even addresses must be used. Using odd addresses to switch to a low-power consumption mode may result in a malfunction.
- To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode, watch mode or timebase timer mode, disable the output of peripheral functions, and set the STP bit of the LPMCR register to 1 or set the TMD bit of the LPMCR register to 0.

Table 8.3-2 Instructions to be Used for Switching to a Low-power Consumption Mode

MOV io,#imm8	MOV dir,#imm8	MOV eam,#imm8	MOV eam,Ri
MOV io,A	MOV dir,A	MOV addr16,A	MOV eam,A
MOV @Rli+disp8,A			
MOVW io,#imm16	MOVW dir,#imm16	MOVW eam,#imm16	MOVW eam,RWi
MOVW io,A	MOVW dir,A	MOVW addr16,A	MOVW eam,A
MOVW @Rli+disp8,A			
SETB io:bp	SETB dir:bp	SETB addr16:bp	
CLRB io:bp	CLRB dir:bp	CLRB addr16:bp	

8.4 CPU Intermittent Operation Mode

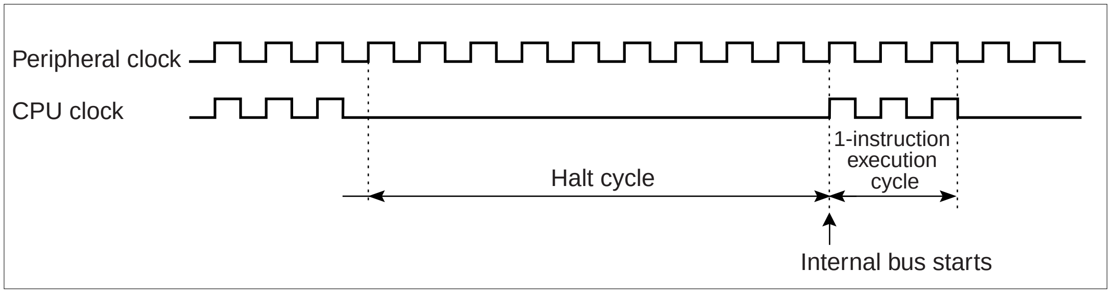
This mode is used for intermittent operation of the CPU while operation clock is supplied to the CPU and peripheral functions. The purpose of this mode is to reduce power consumption.

■ CPU Intermittent Operation Mode

This mode halts the supply of the clock pulse to the CPU for a certain period. The halt occurs after the execution of every instruction that accesses a register, internal memory, I/O, peripheral functions, or the external bus. Internal bus cycle activation is therefore delayed. While high-speed peripheral clock pulses are supplied to peripheral functions, the execution speed of the CPU is reduced, thereby enabling low-power consumption processing.

- The low-power consumption mode control register (LPMCR: CG1 and CG0) is used to select the number of machine cycles that halts the clock supplied to the CPU.
- Instruction execution time in the CPU intermittent operation mode can be calculated. A correction value should be obtained by multiplying the execution count of instructions that access a register, internal memory, internal peripheral functions, or the external bus by the number of clock pulses per halt cycle. Add this corrective value to the normal execution time. Figure 8.4-1 shows the operating clock pulses during the CPU intermittent operation mode.

Figure 8.4-1 Clock Pulses during the CPU Intermittent Operation Mode



8.5 Standby Mode

The standby mode causes the standby control circuit to either stop supplying an operation clock to the CPU or peripheral functions or to stop the oscillation clock reducing power consumption.

■ Operation Status during Standby Mode

Table 8.5-1 shows operation status during standby mode.

Table 8.5-1 Operation Status during Standby Mode

Mode name		Transition conditions	Oscillation clock (HCLK)	Sub-clock (SCLK)	Machine clock	CPU	Peripheral function	Pin	Release method
Sleep mode	Main sleep mode	MCS=1 SCS=1 SLP=1	❍	❍	❍	×	❍	❍	External reset or interrupt
	Sub-sleep mode	MCS=X SCS=0 SLP=1	×	❍	❍	×	❍	❍	External reset or interrupt
	PLL sleep mode	MCS=0 SCS=1 SLP=1	❍	❍	❍	×	❍	❍	External reset or interrupt
Timebase timer mode	SPL=0	MCS=X SCS=1 TMD=0	❍	❍	×	×	×[*1]	◆	External reset or interrupt[*4]
	SPL=1	MCS=X SCS=1 TMD=0	❍	❍	×	×	×[*1]	Hi-Z [*3]	External reset or interrupt[*4]
Watch mode	SPL=0	MCS=X SCS=0 TMD=0	×	❍	×	×	×[*2]	◆	External reset or interrupt[*5]
	SPL=1	MCS=X SCS=0 TMD=0	×	❍	×	×	×[*2]	Hi-Z [*3]	External reset or interrupt[*5]
Stop mode	SPL=0	STP=1	×	×	×	×	×	◆	External reset or interrupt[*6]
	SPL=1	STP=1	×	×	×	×	×	Hi-Z [*3]	External reset or interrupt[*6]

❍: operation, ✕: stop, ◆: held in the state before transiting, Hi-Z: High impedance
*1 : The timebase timer, watch timer, and low voltage detection function operate.
*2 : The watch timer operates.
*3 : The DTP/external interrupt input pin operates.
*4 : Watch timer, timebase timer, and external interrupts
*5 : Watch timer and external interrupts
*6 : External interrupt
MCS: PLL clock select bit in clock selection register (CKSCR)
SCS : Subclock select bit in the clock selection register (CKSCR)
SPL : Pin state specification bit of low-power consumption mode control register (LPMCR)
SLP : Sleep mode bit of low-power consumption mode control register (LPMCR)
STP : Stop mode bit of low-power consumption mode control register (LPMCR)
TMD: Watch mode bit of low-power consumption mode control register (LPMCR)

Note:

For those external pins shared between port functions and peripheral functions in the stop mode, watch mode, or timebase timer mode, disable output for the peripheral functions then set the STP bit to "1" or reset the TMD bit to "0" to set these pins in high impedance state.

8.5.1 Sleep Mode

This mode causes the CPU operating clock to stop during operation in each clock mode. The CPU stops, and peripheral function operates.

■ Switching to Sleep Mode

Writing 1 in the SLP bit and 0 in the STP bit of the low-power consumption mode control register (LPMCR) triggers a switch to a sleep mode according to setting of the MCS and SCS bits in the clock selection register (CKSCR). Table 8.5-2 shows the correspondence between MCS and SCS bits in the clock selection register (CKSCR) and sleep mode.

Table 8.5-2 Setting of Clock Selection Register (CKSCR) and Sleep Mode

Clock selection register (CKSCR)		Sleep mode to be switched
MCS	SCS	
1	1	Main sleep mode
0	1	PLL sleep mode
1	0	Sub-sleep mode
0	0	

Note:

When 1 is written to the SLP and STP bits of the low-power consumption mode control register (LPMCR) at the same time, the STP bit setting overrides the SLP bit setting and the mode switches to the stop mode. When 1 is written to the SLP bit and 0 is written to the TMD bit at the same time, the TMD bit setting overrides the SLP bit setting and the mode switches to the timebase timer mode or watch mode.

● Data retention function

In a sleep mode, the contents of dedicated registers, such as accumulators, and the internal RAM are retained.

● Operation during an interrupt request

Writing 1 in the SLP bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to a sleep mode. If the CPU does not accept the interrupt request, the CPU executes the next to currently executing instruction. If the CPU accepts the interrupt request, CPU operation immediately branches to the interrupt processing routine.

● Status of pins

During a sleep mode, all pins retain their previous status.

■ Return from Sleep Mode

The sleep mode is cancelled by a reset factor or when an interrupt is generated.

● Return by reset factor

When the sleep mode is cancelled by a reset factor, the mode transits to the main clock mode after the sleep mode is cancelled, transiting to the reset sequence.

Note:

- For returning from subsleep mode to main clock mode using the external reset pin ($\overline{\text{RST}}$ pin), input the Low level for at least oscillator's oscillation time* + 100 µs + 16 machine cycles (main clock).
  *: The oscillation time for the oscillator is the period of time taken until its amplitude reaches 90%. It takes several to dozens of ms for crystal oscillators, hundreds of µs to several ms for FAR/ ceramic oscillators, and 0 ms for external clocks.

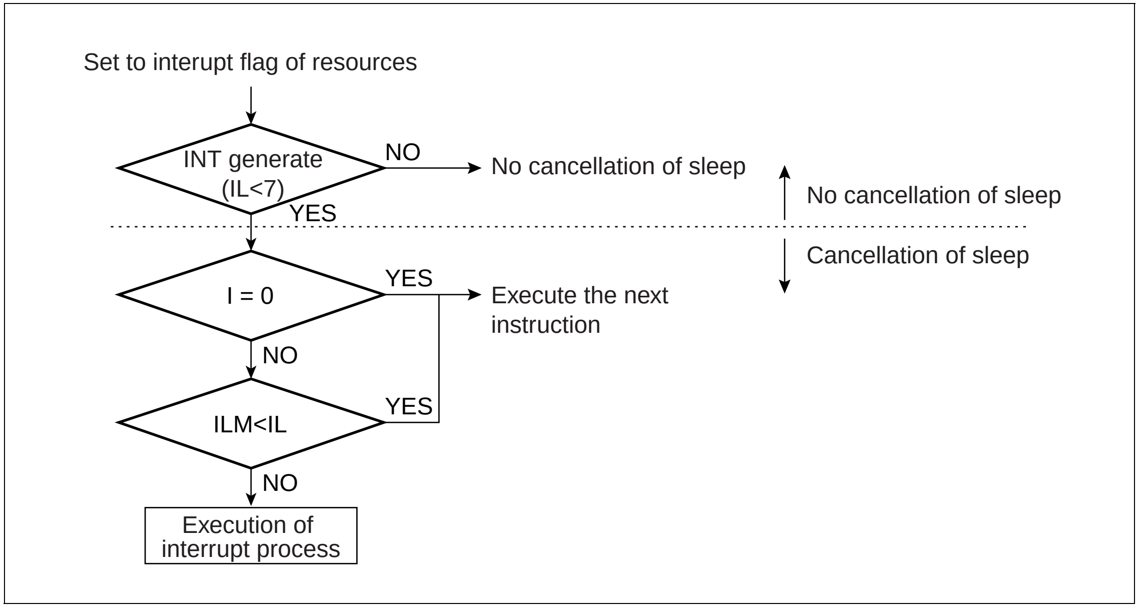
● Return by interrupt

When a higher interrupt request than the interrupt level (IL) of 7 is generated from the resources in the sleep mode, the sleep mode is cancelled. After the sleep mode is cancelled, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR).

- When the CPU is not ready to accept any interrupt request, the next instruction to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.

Figure 8.5-1 shows the release of a sleep mode when an interrupt occurs.

Figure 8.5-1 Release of Sleep Mode by Interrupt Occurrence



Note:

When interrupt processing is executed, the CPU normally executes the instruction that follows the instruction in which a sleep mode has been specified. The CPU then proceeds to interrupt processing.

8.5.2 Watch Mode

This mode causes all functions, excluding the subclock (SCLK), watch timer, and low voltage detection circuit, to stop. Main clock and PLL clock stop.

■ Switching to the Watch Mode

When 0 is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the subclock run mode, switching to the watch mode occurs.

● Data retention function

In the watch mode, the contents of the dedicated registers, such as accumulators, and the internal RAM are retained.

● Operation during an interrupt request

Writing 1 in the TMD bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the watch mode.

If the CPU is not ready to accept any interrupt request, the instruction next to currently executing instruction is executed. If the CPU is ready to accept any interrupt request, an interrupt operation immediately branches to the interrupt processing routine.

● Status of pins

Whether the I/O pins in the watch mode retain the state they had immediately before switching to the watch mode or go to the high-impedance state can be controlled by the SPL bit of the low-power consumption mode control register (LPMCR).

Note:

To set the pin that is shared the peripheral function and port to the high impedance in the watch mode, disable the output of peripheral function, then set the TMD bit of the low-power consumption mode control register (LPMCR) to "0".

■ Return from Watch Mode

The watch mode is cancelled by a reset factor or when an interrupt is generated.

● Return by reset factor

When the watch mode is cancelled by a reset factor, the mode transits to the main clock mode after the watch mode is cancelled, transiting to the reset sequence.

● Return by interrupt

When an interrupt request higher than the interrupt level (IL) of 7 is generated from the watch timer and external interrupt in the watch mode, the watch mode is cancelled. After the watch mode is cancelled, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR). In the sub-watch mode, no oscillation stabilization wait time is generated and the interrupt request is identified immediately after return from the watch mode.

- When the CPU is not ready to accept any interrupt request, the next instruction to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.

Note:

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which the watch mode has been specified. The CPU then proceeds to interrupt processing.

8.5.3 Timebase Timer Mode

This mode causes all functions, excluding oscillation clock (HCLK), subclock (SCLK), the timebase timer, the watch timer, and low voltage detection circuit, to stop. In this mode, only the timebase timer, watch timer, and low voltage detection circuit, operate.

■ Switching to the Timebase Timer Mode

When 0 is written to the TMD bit of the low-power consumption mode control register (LPMCR) in the PLL clock mode or main clock mode (CKSCR: SCM = 1), switching to the timebase timer mode occurs.

● Data retention function

In the timebase timer mode, the contents of dedicated registers, such as accumulators, and the internal RAM are retained.

● Operation during an interrupt request

Writing 0 in the TMD bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the timebase timer mode.

If the CPU is not ready to accept any interrupt request, the instruction next to currently executing instruction is executed. If the CPU is ready to accept any interrupt request, an interrupt operation immediately branches to the interrupt processing routine.

● Status of pins

Whether the I/O pins in the timebase timer mode retain the state they had immediately before switching to the timebase timer mode or go to the high-impedance state can be controlled by the low-power consumption mode control register (LPMCR: SPL).

Note:

To set the pin that is shared the peripheral function and port to the high impedance in the timebase timer mode, disable the output of the peripheral function, then set the TMD bit of the low-power consumption mode control register (LPMCR) to "0".

■ Return from Timebase Timer Mode

The timebase timer mode is cancelled by a reset factor or when an interrupt is generated.

● Return by reset factor

When the timebase timer mode is cancelled by a reset factor, the mode transits to the main clock mode after the timebase timer mode is cancelled, transiting to the reset sequence.

● Return by interrupt

When an interrupt request higher than interrupt level (IL) of 7 is generated from the watch timer, timebase timer, and external interrupt in the timebase timer mode, the timebase timer mode is cancelled. After the timebase timer mode is cancelled, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR).

- When the CPU is not ready to accept any interrupt request, the next instruction to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.
- The following two timebase timer modes are available:
  - Main clock ←→ timebase timer mode
  - PLL clock ←→ timebase timer mode

Note:

When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which switching to the timebase timer mode has been specified. The CPU then proceeds to interrupt processing.

8.5.4 Stop Mode

Because this mode causes oscillation clock (HCLK) and subclock (SCLK) to stop during operation in each clock mode, data can be retained by the lowest power consumption.

■ Stop Mode

When 1 is written to the STP bit of the low-power consumption mode control register (LPMCR) during operation in the PLL clock mode (CKSCR: MCS=1, SCS=0), the mode transits to the stop mode according to the settings of the MCS bit and SCS bit in the clock selection register (CKSCR).

Table 8.5-3 shows the settings of the MCS and SCS bits in the clock selection register (CKSCR) and the stop modes.

Table 8.5-3 Clock Selection Register (CKSCR) Settings and Stop Modes

Clock selection register (CKSCR)		Stop Mode to be Transited
MCS	SCS	
1	1	Main stop mode
0	1	PLL stop mode
1	0	Sub-stop mode
0	0	

Note:

If both the STP and SLP bits in the low-power consumption mode control register (LPMCR) are set to "1" simultaneously, the STP bit is preferred and the mode transits to the stop mode.

● Data retention function

In the stop mode, the contents of the dedicated registers, such as accumulators, and the internal RAM are retained.

● Operation during an interrupt request

Writing 1 in the STP bit of the low-power consumption mode control register (LPMCR) during an interrupt request does not trigger a switch to the stop mode.

If the CPU is not ready to accept any interrupt request, the instruction next to currently executing instruction is executed. If the CPU is ready to accept any interrupt request, an interrupt operation immediately branches to the interrupt processing routine.

● Status of pins

Whether the I/O pins in the stop mode retain the state they had immediately before switching to the stop mode or go to the high-impedance state can be controlled by the SPL bit of the low-power consumption mode control register (LPMCR).

Note:

For those external pins shared between port functions and peripheral functions, disable output for the peripheral functions then set the STP bit to "1" to set these pins in high impedance state.

■ Return from Stop Mode

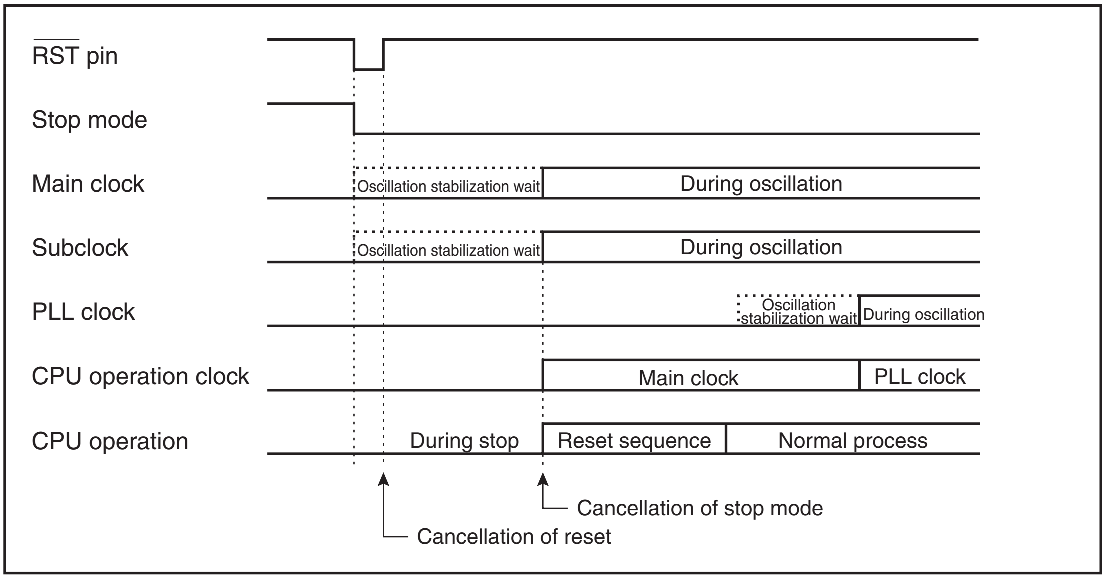
The stop mode is cancelled by a reset factor or when an interrupt is generated. At return from the stop mode, the oscillation clock (HCLK) and the sub clock (SCLK) stop, so the stop mode is cancelled after the elapse of the main clock oscillation stabilization wait time or the sub clock oscillation stabilization wait time.

● Return by reset factor

When the stop mode is cancelled by a reset factor, the main clock oscillation stabilization wait time is generated. After the termination of the main clock oscillation stabilization wait time, the stop mode is cancelled, transiting to the reset sequence.

Figure 8.5-2 shows the return from the sub-stop mode by an external reset.

Figure 8.5-2 Return from the Sub-stop Mode by an External Reset



● Return by interrupt

When an interrupt request higher than the interrupt level (IL) of 7 is generated from external interrupt in the stop mode, the stop mode is cancelled. In the stop mode, the main clock oscillation stabilization wait time or the sub clock oscillation stabilization wait time is generated after the stop mode is cancelled. After the termination of the main clock oscillation stabilization wait time or subclock oscillation stabilization wait time, as with normal interrupt processing, the generated interrupt request is identified according to the settings of the I flag in the condition code register (CCR), the interrupt level mask register (ILM), and the interrupt control register (ICR).

- When the CPU is not ready to accept any interrupt request, the instruction next to the currently executing instruction is executed.
- When the CPU is ready to accept any interrupt request, it immediately branches to the interrupt processing routine.

Notes:

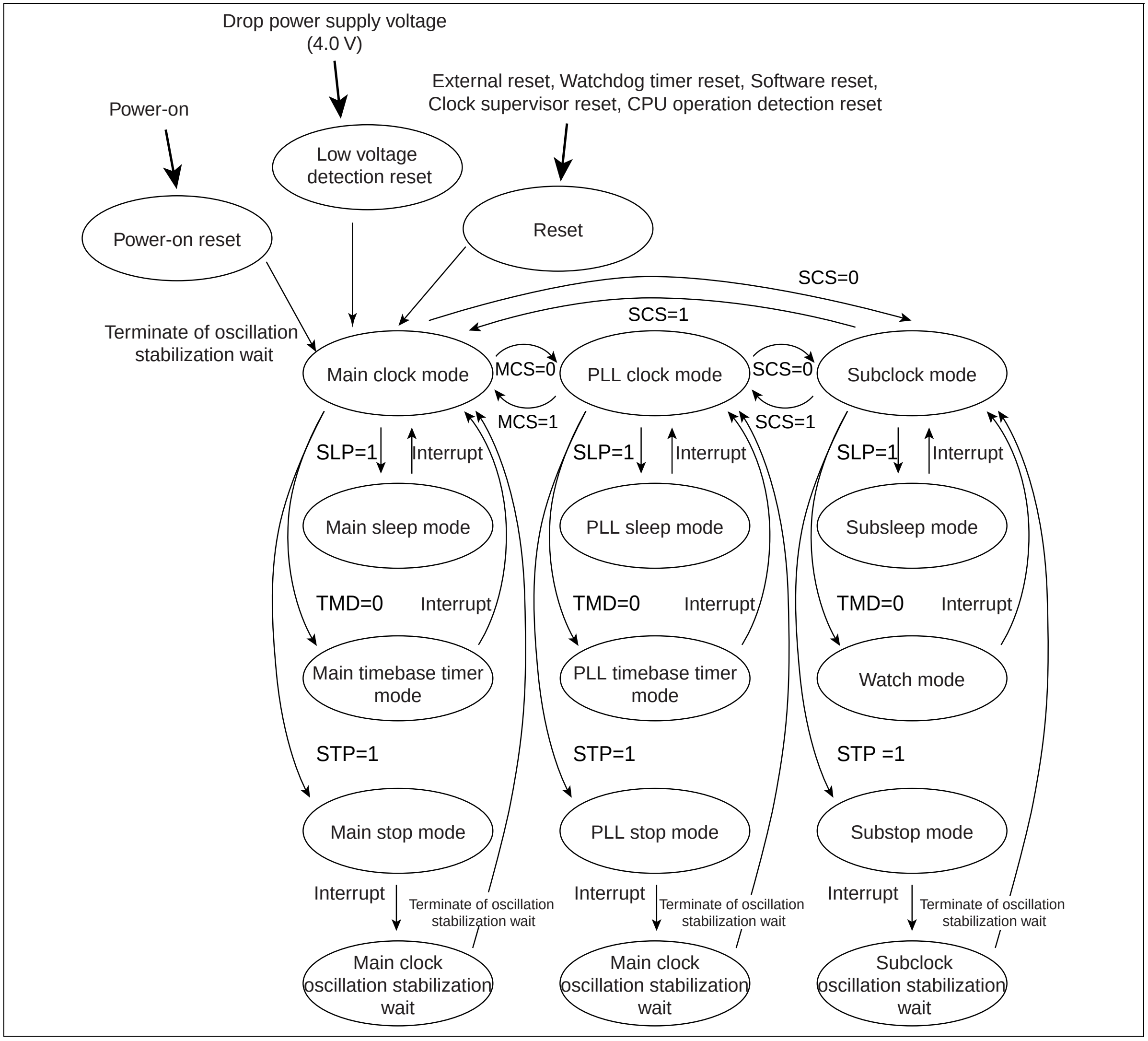
- When interrupt processing is executed, the CPU normally executes the instruction following the instruction in which the stop mode has been specified. The CPU then proceeds to interrupt processing.
  When transiting to the PLL stop mode, set the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) to "10_B" or "11_B".
- In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait times for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0). The oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) must be selected accordingly to account for the longer of main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires 2^{14}/HCLK or more. Set the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) to "10_B" or "11_B".

8.6 Status Change Diagram

Figure 8.6-1 shows the operation status and status transition in the clock mode and standby mode of the MB90360 series.

■ Status Change Diagram

Figure 8.6-1 Status Change Diagram


Drop power supply voltage
(4.0 V)
Power-on
External reset, Watchdog timer reset, Software reset,
Clock supervisor reset, CPU operation detection reset
Low voltage detection reset
Power-on reset
Reset
SCS=0
SCS=1
Terminate of oscillation stabilization wait
Main clock mode
MCS=0
PLL clock mode
SCS=0
Subclock mode
MCS=1
SCS=1
SLP=1
Interrupt
SLP=1
Interrupt
SLP=1
Interrupt
Main sleep mode
PLL sleep mode
Subsleep mode
TMD=0
Interrupt
TMD=0
Interrupt
TMD=0
Interrupt
Main timebase timer mode
PLL timebase timer mode
Watch mode
STP=1
STP=1
STP =1
Main stop mode
PLL stop mode
Substop mode
Interrupt
Terminate of oscillation stabilization wait
Interrupt
Terminate of oscillation stabilization wait
Interrupt
Terminate of oscillation stabilization wait
Main clock oscillation stabilization wait
Main clock oscillation stabilization wait
Subclock oscillation stabilization wait

8.7 Status of Pins in Standby Mode and during Hold and Reset

The status of I/O pins in the standby mode and during hold and reset are described for each memory access mode.

■ Status of I/O Pins (Single-chip Mode)

Table 8.7-1 Status of I/O Pins (Single-chip Mode)

Pin Name	At sleep	At stop/watch/timebase timer		At a reset
		SPL=0	SPL=1	
P27 to P20 P44, P43, P41, P40 P53 to P50 P67 to P60 P87 to P85, P83	Immediately preceding state held[*2]	Input cut off[*4]/ immediately preceding state held[*2]	Input cut off[*4]/ output Hi-Z[*5]	Input disabled[*3]/ output Hi-Z[*5]
P42[*7] P54 to P57 P80, P82, P84[*7]		Input enabled[*1]		

*1: Input enabled means that input function can be used. When the pin is set as input port, handle the pull-up/pull-down or input the external signal. When the pin is set as output port, the pin is set to the same state as other pins.

*2: Indicates that either the output pins output their state as it is immediately before entering each standby mode or the input pins are input-disabled. Output of the output state as it is means that when the resource with an output is in operation, the state of pins is output according to the state of the resource and, when the state of output pins is output, it is held.

*3: Input disabled means that no pin value can be accepted internally because the internal circuit is off while the operation of the input gates of pins is enabled.

*4: Input cut off means that the operation of the input gates of pins is disabled.

*5: Output Hi-Z means that the driving of pin driving transistors is disabled to place the pins in a high impedance state.

*6: In these modes, the pull-up function of the port 2 is invalid.

*7: When the INTxR bit of the external interrupt cause selection register (EISSR) is set to "1", these pins become "input enabled" in stop mode. When the bit is set to "0", they become the same state as other pins.

Note:

To set that pin to high impedance which serves either for a peripheral resource or as a port in stop mode, watch mode, or timebase timer mode, disable the output of the peripheral resource, then set the STP bit to "1" or set the TMD bit to "0" in the low-power consumption mode control register (LPMCR).

8.8 Usage Notes on Low-Power Consumption Mode

This section explains the notes when using the low-power consumption modes.

■ Transition to Standby Mode

When an interrupt request is generated from the resource to the CPU, the mode does not transit to each standby mode even after setting the STP and SLP bits to 1 and the TMD bit to 0 in the low-power consumption mode control register (LPMCR) (and also even after interrupt processing).

If the CPU is servicing an interrupt, the interrupt-service-time interrupt request flag is cleared and the CPU can enter the standby mode unless any other interrupt request has been generated.

■ Notes on the Transition to Standby Mode

To set a pin to high impedance when the pin is shared by a peripheral function and a port in stop mode, watch mode, or timebase timer mode, use the following procedure:

1. Disable the output of peripheral functions.
2. Set the SPL bit to "1", STP bit to "1", or TMD bit to "0" in the low-power consumption mode control register (LPMCR).

■ Cancellation of Standby Mode by Interrupt

When an interrupt request higher than the interrupt level (IL) of 7 is generated from the resource and external interrupt during operation in the sleep mode, watch mode, timebase timer mode, or stop mode, the standby mode is cancelled. The standby mode is cancelled by an interrupt regardless of whether the CPU accepts interrupts or not.

Note:

To prevent the CPU from causing a branch to interrupt servicing immediately after returning from standby mode, take measures, such as disabling interrupts before setting the standby mode.

■ Note on Canceling Standby Mode

The standby mode can be cancelled by an input according to the settings of an input factor of an external interrupt. The input factor can be selected from High level, Low level, rising edge, and falling edge.

■ Oscillation Stabilization Wait Time

● Oscillation stabilization wait time of main clock

In the sub clock mode, watch mode, or stop mode, the oscillation of the main clock stops and the oscillation stabilization wait time of the main clock is required. The oscillation stabilization wait time of the main clock is set by the WS1 and WS0 bits in the clock selection register (CKSCR).

● Oscillation stabilization wait time of sub clock

In the sub-stop mode, the oscillation of the sub clock (SCLK) stops and the oscillation stabilization wait time of the sub clock is required. The oscillation stabilization wait time of the sub clock is fixed at 2^{14}/ SCLK (SCLK: sub clock).

● PLL clock oscillation stabilization wait time

In main clock mode, the PLL multiplication circuit stops. When changing to PLL clock mode, it is necessary to reserve the PLL clock oscillation stabilization wait time. The CPU runs in main clock mode till the PLL clock oscillation stabilization wait time has elapsed. When the main clock mode is switched to PLL clock mode, the PLL clock oscillation stabilization wait time is fixed at 2^{14}/HCLK (HCLK: oscillation clock).

In sub-clock mode, the main clock and PLL multiplication circuit stop. When changing to PLL clock mode, it is necessary to reserve the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait time for main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0). The oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) must be selected accordingly to account for the longer of the main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires 2^{14}/HCLK or more. Set the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) to "10_B" or "11_B".

In PLL stop mode, the main clock and PLL multiplication circuit stop. During recovery from PLL stop mode, it is necessary to allot the main clock oscillation stabilization wait time and PLL clock oscillation stabilization wait time. The oscillation stabilization wait time for the main clock and PLL clock are counted simultaneously according to the value specified in the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0). The oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) must be selected accordingly to account for the longer of main clock and PLL clock oscillation stabilization wait time. The PLL clock oscillation stabilization wait time, however, requires 2^{14}/HCLK or more. Set the oscillation stabilization wait time selection bits in the clock selection register (CKSCR: WS1, WS0) to "10_B" or "11_B".

■ Clock Mode Switching

When the clock mode is switched, do not switch to other clock mode and low-power consumption mode before this switching is completed. Confirm the completion of clock mode switching by referring to the MCM and SCM bits of the clock selection register (CKSCR).

If the mode is switched to other clock mode or low-power consumption mode before completion of switching, the mode may not be switched.

■ Notes on Accessing the Low-Power Consumption Mode Control Register (LPMCR) to Enter the Standby Mode

● To access the low-power consumption mode control register (LPMCR) with assembler language

To set the low-power consumption mode control register (LPMCR) to enter the standby mode, use the instruction listed in Table 8.3-2 .

The standby mode transition instruction in Table 8.3-2 must always be followed by an array of instructions highlighted by a line below.

```
 MOV  LPMCR, #H’ xx; The low-power consumption mode transition instruction in Table 8.3-2
 NOP
 NOP
 JMP  $+3                        ; Jump to the next instruction
 MOV  A, #H’ 10                  ; Arbitrary instruction
```

The devices does not guarantee its operation after returning from the standby mode if you place an array of instructions other than the one enclosed in the line.

● To access the low-power consumption mode control register (LPMCR) with C language

To enter the standby mode using the low-power consumption mode control register (LPMCR), use one of the following methods 1. to 3. to access the register:

1. Specify the standby mode transition instruction as a function and insert two __wait_nop() built-in functions after that instruction. If any interrupt other than the interrupt to return from the standby mode can occur within the function, optimize the function during compilation to suppress the LINK and UNLINK instructions from occurring.

   Example: Watch mode or timebase timer mode transition function

```
void enter_watch(){
    IO_LPMCR.byte = 0x10;  /* Set LPMCR TMD bit to 0 */
    __wait_nop();
    __wait_nop();
}
```

2. Define the standby mode transition instruction using __asm statements and insert two NOP and JMP instructions after that instruction.

   Example: Transition to sleep mode

```
  __asm("MOV I:_IO_LPMCR, #H’ 58);        /* Set LPMCR SLP bit to 1 */
  __asm("NOP");
  __asm("OP");
  __asm("JMP  $+3");                      /* Jump to the next instruction*/
```

3. Define the standby mode transition instruction between #pragma asm and #pragma endasm and insert two NOP and JMP instructions after that instruction.

   Example: Transition to stop mode

```
#pragma asm
          MOV  I:_IO_LPMCR, #H’ 98       /* Set LPMCR STP bit to 1 */
          NOP
          NOP
          JMP   $+3                      /* Jump to the next instruction */
#pragma endasm
```

CHAPTER 9

MEMORY ACCESS MODES

This chapter explains the functions and operations of the memory access modes.

9.1 Outline of Memory Access Modes

In the F^{2}MC-16LX, various modes are provided for access methods and access areas.

■ Outline of Memory Access Modes

Table 9.1-1 Mode Pin and Mode

Operation mode	Bus mode
RUN mode	Single-chip
Flash programming	–

● Operation mode

Operation mode means the mode for controlling the device operation status. The operation mode is specified by the MDx mode setting pin and the Mx bit in mode data. By selecting an operation mode, normal operation activation or flash memory programming can be performed.

● Bus mode

Bus mode means the mode for controlling the internal ROM operation and external access function. The bus mode is specified by the MDx mode setting pin and the Mx bit in mode data. The MDx mode setting pin specifies the bus mode for reading the reset vector and mode data, and the Mx bit in mode data specifies the bus mode for normal operation.

● Run mode

Run mode means the CPU operation mode. The run mode has the main clock mode, PLL clock mode, and various low-power consumption mode. See CHAPTER 8 "LOW-POWER CONSUMPTION MODE" for details.

9.1.1 Mode Pins

Table 9.1-2 lists the operations that can be specified by combining the three external pins MD2 to MD0.

■ Mode Pins

Table 9.1-2 Mode Pin and Mode

Mode pin setting			Mode name	Reset vector access area	External data bus width	Remarks
MD2	MD1	MD0				
0	0	0	Setting disabled			
0	0	1				
0	1	0				
0	1	1	Internal vector mode	Internal	(Mode data)	Reset sequence and subsequent sequences are controlled by mode data.
1	0	0	Setting disabled			
1	0	1				
1	1	0	Flash serial programming*	–	–	–
1	1	1	Flash memory	–	–	Mode when parallel writer is used

*: The serial programming of the flash memory cannot be written only by setting the mode pin. Other pin also need to be set. See "CHAPTER 25 EXAMPLES OF MB90F362/T(S), MB90F367/T(S) SERIAL PROGRAMMING CONNECTION" for details.

9.1.2 Mode Data

Mode data is stored at FFFFDF$_H$ of main memory and used for controlling the CPU operation. This data is fetched during a reset sequence and stored in the mode register inside the device. The mode register value can be changed only by a reset sequence. The setting of this register is valid after the reset sequence.
Always set the reserved bits to "0".

■ Mode Data

Figure 9.1-1 Mode Data Configuration

	7	6	5	4	3	2	1	0
Address: FFFFDF$_H$	M1	M0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

[bit7 and bit6] M1, M0 (bus mode setting bits)

The M1 and M0 bits are used to specify the operation mode after the reset sequence is completed. Table 9.1-3 shows the relationship between the M1 and M0 bits and the functions.

Table 9.1-3 Function of M1, M0 (bus mode setting bit)

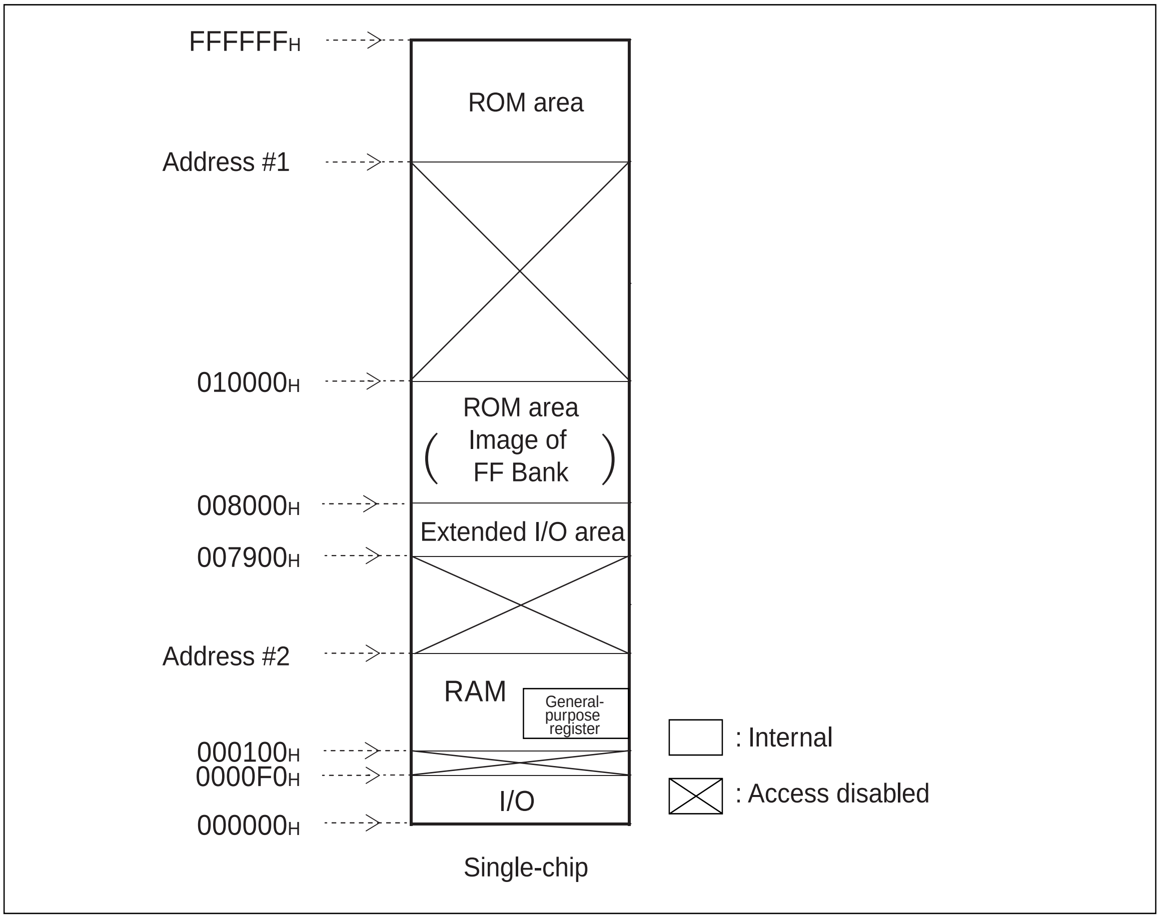
M1	M0	Function	Remarks
0	0	Single-chip mode	
0	1	Setting disabled	
1	0		
1	1		

9.1.3 Memory Space in Each Bus Mode

Figure 9.1-2 shows the correspondence between the access areas and physical addresses for each bus mode.

■ Memory Space in Each Bus Mode

Figure 9.1-2 Relationship between Access Areas and Physical Addresses for Each Bus Mode



Product type	Address #1	Address #2
MB90F362/T(S), MB90362/T(S) MB90F367/T(S), MB90367/T(S)	FF0000$_H$	000D00$_H$
MB90V340A-101/102	F80000$_H$	007900$_H$

■ Recommended Setting

Table 9.1-4 lists an example of recommended settings for mode pins and mode data.

Table 9.1-4 Recommended Setting Example of Mode Pin and Mode Data

Setting example	MD2	MD1	MD0	M1	M0
Single-chip	0	1	1	0	0

External pins have signal functions that depend on each mode.

CHAPTER 10

I/O PORTS

This chapter explains the functions and operations of the I/O ports.

10.1 I/O Ports

Each pin of the ports can be specified as input or output using the port direction register (DDR) if the corresponding peripheral does not use the pin. When a pin is specified as input, the logic level at the pin is read. When a pin is specified as output, the port data register value is read. The above also applies to a read operation for the read-modify-write instructions.

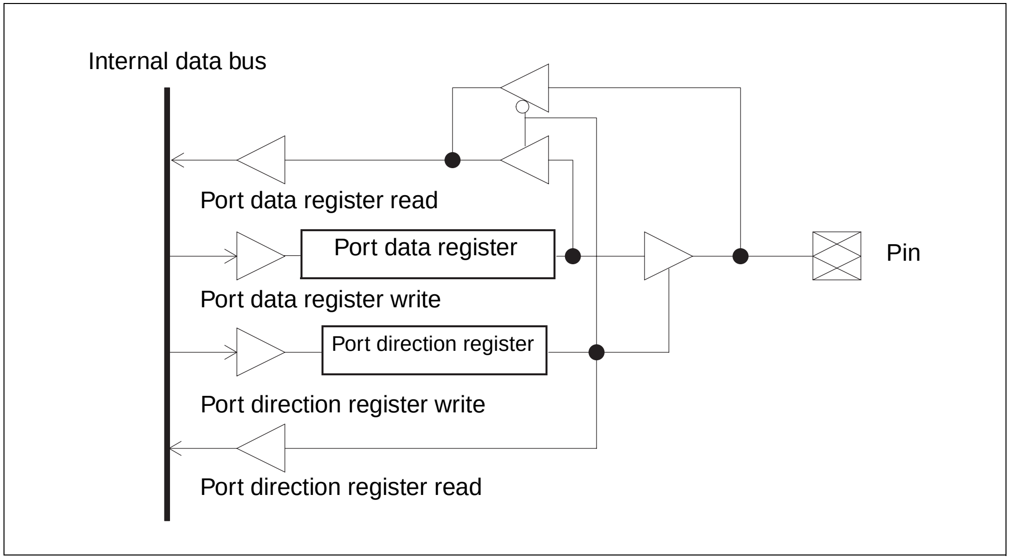
■ I/O Ports

When a pin is used as an output of other peripheral function, the logic level at the pin is read regardless of the port data register value.

It is generally recommended that the read-modify-write instructions should not be used for setting the port data register prior to setting the port as an output. This is because the read-modify-write instruction in this case results reading the logic level at the port rather than the register value.

Figure 10.1-1 is a block diagram of the I/O ports.

Figure 10.1-1 I/O Port Block Diagram



10.2 I/O Port Registers

There are five types of I/O port registers:

- **Port data register (PDR2, PDR4 to PDR6, PDR8)**
- **Port direction register (DDR2, DDR4 to DDR6, DDR8, DDRA)**
- **Pull-up control register (PUCR2)**
- **Analog input enable register (ADER5, ADER6)**
- **Input level select register (ILSR0, ILSR1)**

■ I/O Port Registers

Figure 10.2-1 shows the I/O port registers.

Figure 10.2-1 I/O Port Registers

Bit No. ⇨	7	6	5	4	3	2	1	0	
Address: 000002$_H$	P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address: 000004$_H$	–	–	–	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address: 000005$_H$	P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address: 000006$_H$	P67	P66	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address: 000008$_H$	P87	P86	P85	P84	P83	P82	–	P80	Port 8 data register (PDR8)

Bit No. ⇨	7	6	5	4	3	2	1	0	
Address: 000012$_H$	D27	D26	D25	D24	D23	D22	D21	D20	Port 2 direction register (DDR2)
Address: 000014$_H$	–	–	–	D44	D43	D42	D41	D40	Port 4 direction register (DDR4)
Address: 000015$_H$	D57	D56	D55	D54	D53	D52	D51	D50	Port 5 direction register (DDR5)
Address: 000016$_H$	D67	D66	D65	D64	D63	D62	D61	D60	Port 6 direction register (DDR6)
Address: 000018$_H$	D87	D86	D85	D84	D83	D82	–	D80	Port 8 direction register (DDR8)
Address: 00001A$_H$	–	–	–	SIL1	SIL0	–	–	–	Port A direction register (DDRA)

Bit No. ⇨	7	6	5	4	3	2	1	0	
Address: 00001E$_H$	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20	Port 2 pull-up control register (PUCR2)

Bit No. ⇨	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00000B$_H$	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8	Port 5 analog input enable register (ADER5)
Address: 00000C$_H$	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 6 analog input enable register (ADER6)

Bit No. ⇨	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address: 00000E$_H$	–	IL6	IL5	IL4	–	IL2	–	–	Input level select register (ILSR0)
Address: 00000F$_H$	–	–	–	–	–	–	–	IL8	Input level select register (ILSR1)

10.2.1 Port Data Register (PDR)

Note that R/W for I/O ports differ from R/W for memory in the following points:

- **Input mode**

Read: The level at the corresponding pin is read.

Write: Data is written to an output latch.

- **Output mode**

Read: The port data register latch value is read.

Write: Data is written to an output latch and outputted to the corresponding pin.

Figure 10.2-2 shows the port data registers (PDR).

■ Port Data Register (PDR)

Figure 10.2-2 Port Data Registers (PDR)

	Bit No. ⇨	7	6	5	4	3	2	1	0	Reset value	Access
PDR2	Address: 000002$_H$	P27	P26	P25	P24	P23	P22	P21	P20	Undefined	R/W
PDR4	Address: 000004$_H$	—	—	—	P44	P43	P42	P41	P40	Undefined	R/W
PDR5	Address: 000005$_H$	P57	P56	P55	P54	P53	P52	P51	P50	Undefined	R/W
PDR6	Address: 000006$_H$	P67	P66	P65	P64	P63	P62	P61	P60	Undefined	R/W
PDR8	Address: 000008$_H$	P87	P86	P85	P84	P83	P82	—	P80	Undefined	R/W

● Reading the port data register

The value obtained when reading the port data register (PDR) depends on the status of the port direction register (DDR) and status of the peripheral function connected to the pin.

The following shows the value obtained by each combination.

Value of DDR	Output state of peripheral function	Reading value
0 (input)	Enabled	Output value from peripheral function
1 (output)	Enabled	Output value from peripheral function
0 (input)	Disabled	Pin state
1 (output)	Disabled	Value of output latch

Further, when using as input by peripheral function, set the DDR of the connected pin to "0" (input).

10.2.2 Port Direction Register (DDR)

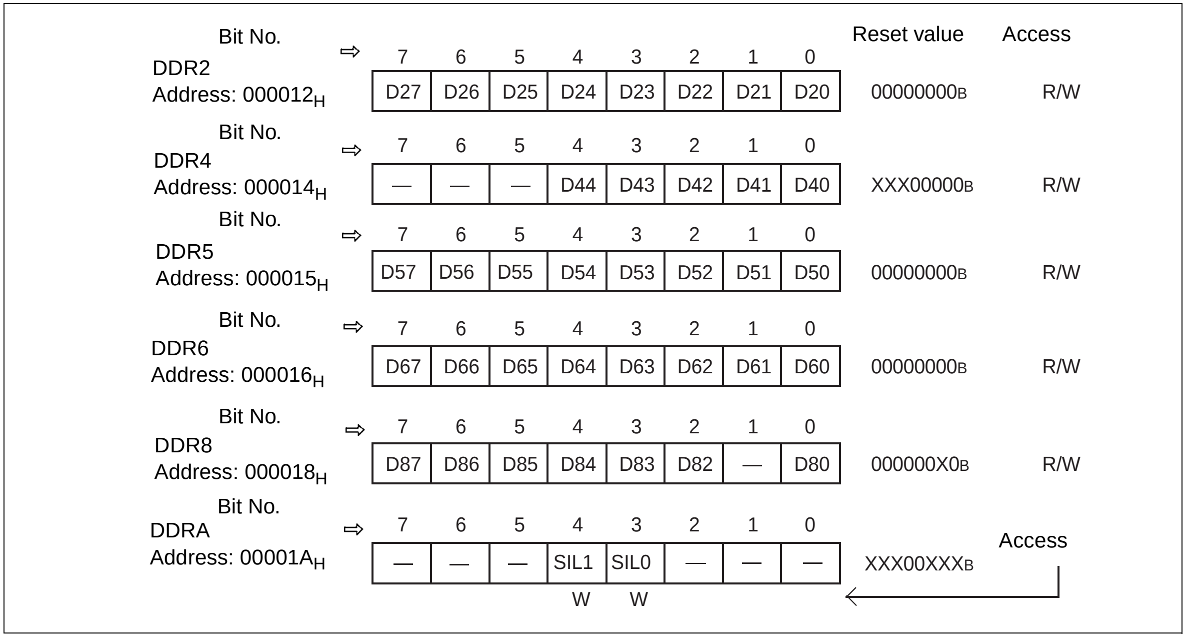
This register has following functions:

- **Setting the data direction of each pin that is used as a port.**
- **Setting the input level of SIN -- Serial data input pin for LIN-UART.**

■ Port Direction Register (DDR)

Figure 10.2-3 shows the Port Direction Registers (DDR).

Figure 10.2-3 Port Direction Registers (DDR)



Bit No. ⇨	7	6	5	4	3	2	1	0	Reset value	Access
DDR2 Address: 000012H	D27	D26	D25	D24	D23	D22	D21	D20	00000000B	R/W
DDR4 Address: 000014H	—	—	—	D44	D43	D42	D41	D40	XXX00000B	R/W
DDR5 Address: 000015H	D57	D56	D55	D54	D53	D52	D51	D50	00000000B	R/W
DDR6 Address: 000016H	D67	D66	D65	D64	D63	D62	D61	D60	00000000B	R/W
DDR8 Address: 000018H	D87	D86	D85	D84	D83	D82	—	D80	000000X0B	R/W
DDRA Address: 00001AH	—	—	—	SIL1	SIL0	—	—	—	XXX00XXXB	
Access				W	W					

Bits Dxx (DDR2, DDR4 to DDR6, DDR8)

These bits set to the I/O direction of the port. When each pin is used as port, the corresponding pin is controlled below.

When set to "0": The corresponding pin is set to input mode.

When set to "1": The corresponding pin is set to output mode.

Bits SIL0, SIL1 (DDRA bit3, bit4)

These bits set the input level of the corresponding SIN (Serial Data Input for LIN-UART) pin forcibly. SIL0 to SIL1 correspond to SIN0 (LIN-UART0) to SIN1(LIN-UART1), respectively.

When setting to “0”: CMOS or Automotive is selected for the input level depending on the setting of the corresponding ILx bit and ILTx bit in the ILSR. (See "10.2.5 Input Level Select Register" for ILSR.)

When set to “1”: CMOS is selected for the input level regardless of the setting of the corresponding ILx bit and ILTx bit in ILSR.

The initial value of these bits is “0”.

Table 10.2-1 SIN0/SIN1 Input Level Setting

DDRA	ILSR	SIN0(P82) / SIN1(P85) input level
SIL0/SIL1 bit	IL8 bit	
0	0	Automotive level
0	1	CMOS level
1	x	CMOS level

Note:

SIL0, SIL1 are write-only, and “1” is always read from these bits. Therefore, instructions that perform a read-modify-write (RMW) operation such as the INC/DEC instruction, cannot be used at DDRA.

- DDRA: Bits 0 to 2, Bits 5 to 7 (unused bits)

"1" is always read from these bits. Writing to these bits is no effect.

10.2.3 Pull-up Control Register (PUCR)

Each pin of port2 has programmable pull-up resistor. Each bit of this register controls corresponding pull-up resistor whether to be used or not.

Figure 10.2-4 shows the pull-up control register (PUCR), and Figure 10.2-5 is the block diagram.

■ Pull-up Control Register (PUCR)

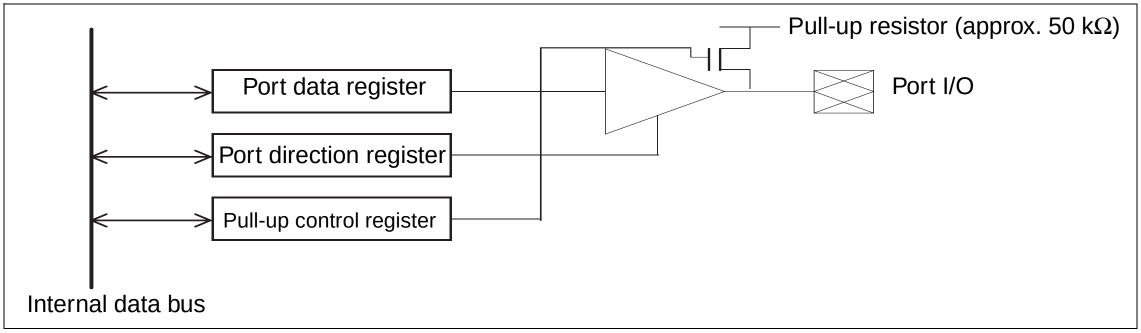
Figure 10.2-4 Pull-up Control Register (PUCR)

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: $00001E_H$	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20	PUCR2
Read/Write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

R/W: Read/Write

■ Block Diagram of Pull-up Control Register (PUCR)

Figure 10.2-5 Block Diagram of Pull-up Control Register (PUCR)



In input mode, the pull-up resistor is controlled.

0: No pull-up resistor in input mode

1: Pull-up resistor in input mode

Note:

In output mode, this register has no meaning (no pull-up resistor).

The port direction register (DDR) determines the input-output mode.

In stop mode (SPL=1), the state with no pull-up resistor is entered (high impedance).

10.2.4 Analog Input Enable Register (ADER)

Figure 10.2-6 shows the analog input enable register.

■ Analog Input Enable Registers (ADER)

Figure 10.2-6 Analog Input Enable Registers (ADER6, ADER5)

ADER6	7	6	5	4	3	2	1	0	Initial value	Access
Address: $00000C_H$	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	11111111_B	R/W

ADER5	15	14	13	12	11	10	9	8		
Address: $00000B_H$	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8	11111111_B	R/W

R/W: Read/Write

Each bit of ADER6/ADER5 sets to enable/disable the analog input of each pin in the port 6 and port 5. ADER6 and ADER5 correspond to the port 6 and port 5, respectively.

When set to "0": The corresponding pin is set to disable the analog input. A pin set to disable the analog input can be used as I/O pin for peripheral function other than I/O port and A/D converter.

When set to "1": The corresponding pin is set to the analog input mode. A pin set to the analog input mode is the dedicated analog input pin for the A/D converter. The pin cannot be used as I/O pin of I/O port and other peripheral function.

Note:

When the analog input enable bit (ADEx) is set to "1", each pin of the port 6 and port 5 is the analog input pin for the A/D converter. Initial value of the ADEx bit is "1". Therefore, the corresponding pins cannot be used as I/O pin of peripheral function other than I/O port and A/D converter at the initial setting. When the pin is used as I/O pin of other peripheral function and I/O port, the ADEx bit is set to "0".

10.2.5 Input Level Select Register

The input level select register allows to switch from Automotive Hysteresis input levels to CMOS Hysteresis input levels.

■ Input Level Select Register (ILSR)

The input level select register ILSR is located on addresses $0E_H$ and $0F_H$.

Figure 10.2-7 Input Level Select Register (ILSR)

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILSR1 : $00000F_H$ ILSR0 : $00000E_H$	–	–	–	–	–	–	–	IL8	–	IL6	IL5	IL4	–	IL2	–	–
	–	–	–	–	–	–	–	R/W	–	R/W	R/W	R/W	–	R/W	–	–
Initial value:	X	X	X	X	X	X	X	0/1	X	0/1	0/1	0/1	X	0/1	X	X

R/W : Read/Write
– : Unused
X : Undefined

bit 2, bit 4 to bit6 and bit8:IL2, IL4 to IL6, IL8

These bits set the input level of the corresponding port. IL2, IL4 to IL6, IL8 correspond to the port 2, port 4 to port 6, port 8, respectively. Setting these bits to “0” selects the Automotive input level. Setting these bits to “1” selects the CMOS hysteresis input level. The initial value of these bits depends on the mode pin setting:

- For the flash memory mode, the initial value is “1” (TTL).
- For all other modes, the initial value is "0" (Automotive).

bit 0, bit 1, bit 3, bit 7 and bit 9 to bit 15: undefined

Reading from these bits is undefined. Writing to these bits is no effect.

Note:

The threshold of the corresponding input pin varies immediately after the setting of the input level select register is changed.

Therefore, do not use the read value from the pin until 2 machine cycles are elapsed after the setting is changed.

When the setting is changed, be sure to disable the corresponding resource.

■ Initial value of ILSR

Initial value of each bit of ILSR is determined when external reset signal is released depending on the value of MD2, MD1, MD0 pin input, as shown in following table.

About detail of each mode, please see "CHAPTER 9 MEMORY ACCESS MODES".

Table 10.2-2 Relationship between Mode Pin and Initial Value of Input Level Select Register (ILSR)

MD2	MD1	MD0	Operation mode	Initial value	Port input level
				ILx	Port 2, 4 to 6, 8
0	0	0	Reserved		
0	0	1			
0	1	0			
0	1	1	Internal vector mode	0	Automotive
1	0	0	Reserved		
1	0	1			
1	1	0	Flash serial write	0	Automotive
1	1	1	Flash memory	1	TTL

CHAPTER 11

TIMEBASE TIMER

This chapter explains the functions and operations of the timebase timer.

11.1 Overview of Timebase Timer

The timebase timer is an 18-bit free-run counter (timebase timer counter) that increments in synchronization with the main clock (half frequency of main oscillation clock).

- **Four interval times can be selected and an interrupt request can be generated for each interval time.**
- **An operation clock is supplied to the oscillation stabilization wait time timer and other peripherals.**

■ Interval Timer Function

- When the timebase timer counter reaches the interval time set by the interval time select bits (TBTC: TBC1, TBC0), an overflow (carrying) occurs (TBTC: TBOF = 1) and an interrupt request is generated.
- When an interrupt is enabled when an overflow occurs (TBTC: TBIE = 1), an overflow occurs (TBTC: TBOF = 1) and an interrupt is generated.
- The timebase timer has four interval times that can be selected. Table 11.1-1 shows the interval times of the timebase timer.

Table 11.1-1 Interval Times of Timebase Timer

Count clock	Interval time
2/HCLK(0.5 µs)	2^{12}/HCLK (approx. 1.0 ms)
	2^{14}/HCLK (approx. 4.1 ms)
	2^{16}/HCLK (approx. 16.4 ms)
	2^{19}/HCLK (approx. 131.1 ms)

HCLK: Oscillation clock
The parenthesized values are provided at 4-MHz oscillation clock.

■ Clock Supply

The timebase timer supplies an operation clock to the resources such as an oscillation stabilization wait time timer, PPG timer, and watchdog timer. Table 11.1-2 shows the clock cycles supplied from the timebase timer to each resource.

Table 11.1-2 Clock Cycles Supplied from Timebase Timer

Where to supply clock	Clock cycle
Oscillation stabilization wait time*	2^{10}/HCLK (approx. 256 µs)
	2^{13}/HCLK (approx. 2.0 ms)
	2^{15}/HCLK (approx. 8.2 ms)
	2^{17}/HCLK (approx. 32.8 ms)
Watchdog timer	2^{12}/HCLK (approx. 1.0 ms)
	2^{14}/HCLK (approx. 4.1 ms)
	2^{16}/HCLK (approx. 16.4 ms)
	2^{19}/HCLK (approx. 131.1 ms)
PPG timer	2^{9}/HCLK (approx. 128 µs)

HCLK: Oscillation clock
The parenthesized values are provided at 4-MHz oscillation clock.
*:As the oscillation cycle is unstable immediately after oscillation starts, standard oscillation stabilization wait time values are given as a guide.

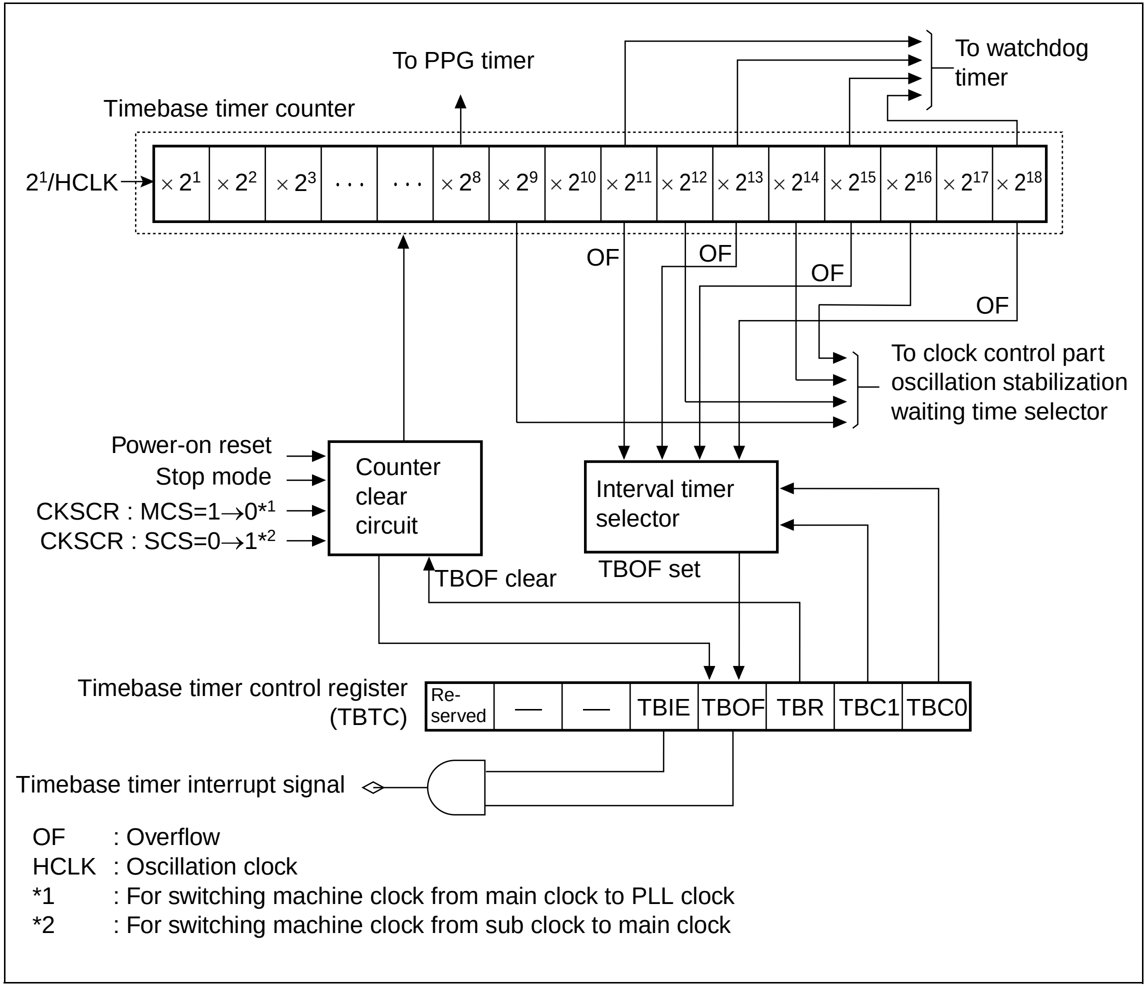
11.2 Block Diagram of Timebase Timer

The timebase timer consists of the following blocks:

- **Timebase timer counter**
- **Counter clear circuit**
- **Interval timer selector**
- **Timebase timer control register (TBTC)**

■ Block Diagram of Timebase Timer

Figure 11.2-1 Block Diagram of Timebase Timer



The actual interrupt request number of the timebase timer is as follows:

Interrupt request number: #25 (19_H)

● Timebase timer counter

The timebase timer counter is an 18-bit up counter that uses a clock with a half frequency of the oscillation clock (HCLK) as a count clock.

● Counter clear circuit

The counter clear circuit clears the value of the timebase timer counter by the following factors:

- Timebase timer counter clear bit in the timebase timer control register (TBTC: TBR=0)
- Power-on reset
- Transition to main stop mode or PLL stop mode (CKSCR:SCS=1, LPMCR: STP=1)
- Switching the clock mode (from main clock mode to PLL clock mode, from subclock mode to PLL clock mode, or from subclock mode to main clock mode)

● Interval timer selector

The interval timer selector selects the output of the timebase timer counter from four types.
When incrementing causes the selected interval time bit to overflow (carrying), an interrupt request is generated.

● Timebase timer control register (TBTC)

The timebase timer control register (TBTC) selects the interval time, clears the timebase timer counter, enables or disables interrupts, and checks and clears the state of an interrupt request.

11.3 Configuration of Timebase Timer

This section explains the registers and interrupt factors of the timebase timer.

■ List of Registers and Reset Values of Timebase Timer

Figure 11.3-1 List of Registers and Reset Values of Timebase Timer

	bit	15	14	13	12	11	10	9	8
Timebase timer control register (TBTC)		1	×	×	0	0	1	0	0

× : Undefined

■ Generation of Interrupt Request from Timebase Timer

When the selected interval timer counter bit reaches the interval time, the overflow interrupt request flag bit in the timebase timer control register (TBTC: TBOF) is set to "1". If the overflow interrupt request flag bit is set (TBTC: TBOF = 1) when the interrupt is enabled (TBTC: TBIE = 1), the timebase timer generates an interrupt request.

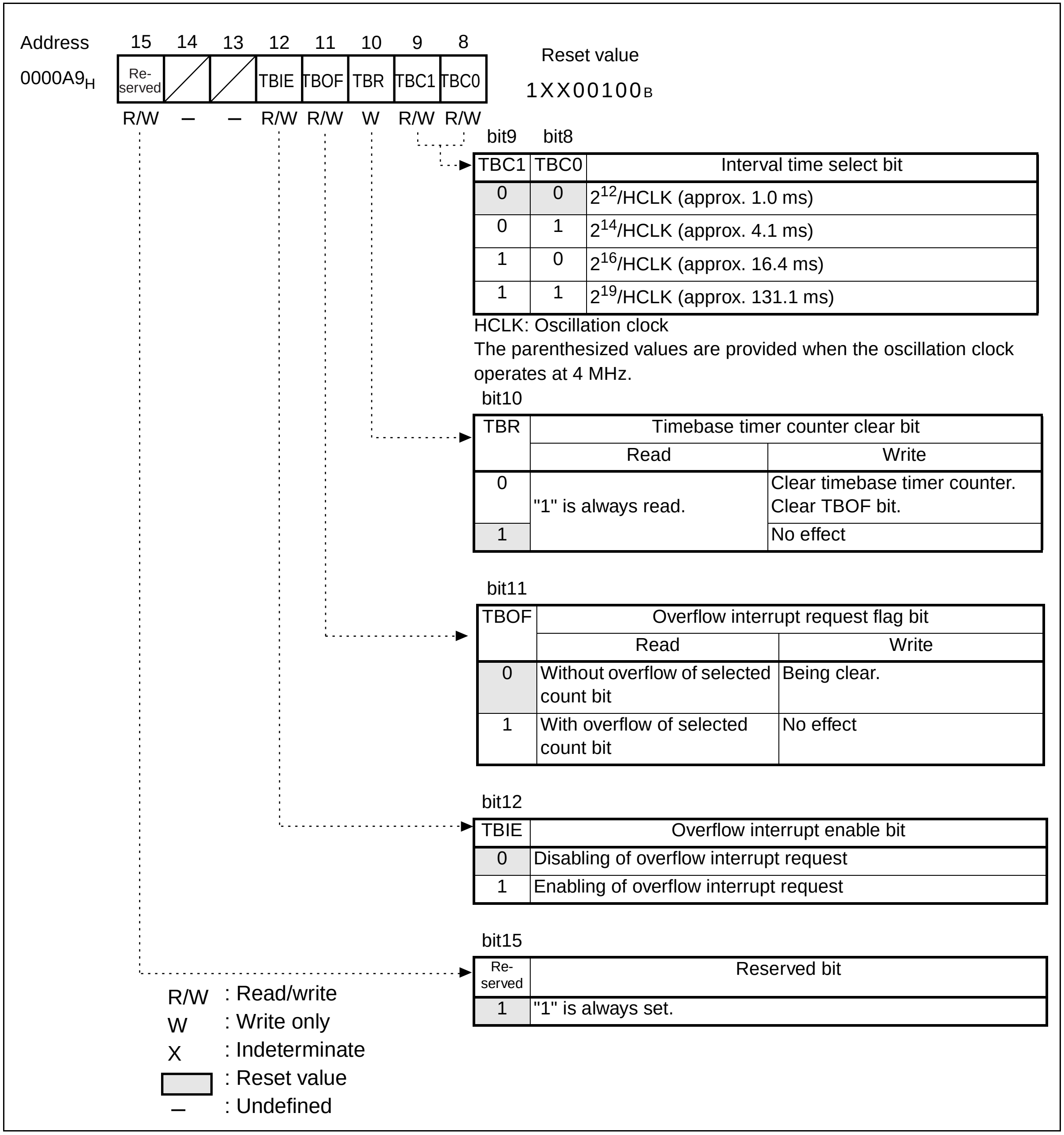
11.3.1 Timebase timer control register (TBTC)

The timebase timer control register (TBTC) provides the following settings:

- **Selecting the interval time of the timebase timer**
- **Clearing the counter value of the timebase timer**
- **Enabling or disabling the interrupt request when an overflow occurs**
- **Checking and clearing the state of the interrupt request flag when an overflow occurs**

■ Timebase Timer Control Register (TBTC)

Figure 11.3-2 Timebase Timer Control Register (TBTC)



Address	15	14	13	12	11	10	9	8	Reset value
$0000A9_H$	Re-served			TBIE	TBOF	TBR	TBC1	TBC0	1XX00100B
	R/W	–	–	R/W	R/W	W	R/W	R/W	

bit9 bit8

TBC1	TBC0	Interval time select bit
0	0	2^{12}/HCLK (approx. 1.0 ms)
0	1	2^{14}/HCLK (approx. 4.1 ms)
1	0	2^{16}/HCLK (approx. 16.4 ms)
1	1	2^{19}/HCLK (approx. 131.1 ms)

HCLK: Oscillation clock
The parenthesized values are provided when the oscillation clock operates at 4 MHz.

bit10

TBR	Timebase timer counter clear bit	
	Read	Write
0	"1" is always read.	Clear timebase timer counter. Clear TBOF bit.
1		No effect

bit11

TBOF	Overflow interrupt request flag bit	
	Read	Write
0	Without overflow of selected count bit	Being clear.
1	With overflow of selected count bit	No effect

bit12

TBIE	Overflow interrupt enable bit
0	Disabling of overflow interrupt request
1	Enabling of overflow interrupt request

bit15

Re-served	Reserved bit
1	"1" is always set.

R/W	: Read/write
W	: Write only
X	: Indeterminate
▢	: Reset value
–	: Undefined

Table 11.3-1 Functions of Timebase Timer Control Register (TBTC)

Bit name		Function
bit15	Reserved: reserved bit	Always set this bit to "1".
bit14 bit13	Undefined bits	**Read:** The value is undefined. **Write:** No effect
bit12	TBIE: Overflow interrupt enable bit	This bit enables or disables an interrupt when the interval timer bit in the timebase timer counter overflows. **When set to "0":** No interrupt request is generated at an overflow (TBOF = 1). **When set to "1":** An interrupt request is generated at an overflow (TBOF = 1).
bit11	TBOF: Overflow interrupt request flag bit	This bit indicates an overflow (carrying) in the interval timer bit in the timebase timer counter. When an overflow (carrying) occurs (TBOF = 1) with interrupts enabled (TBIE = 1), an interrupt request is generated. **When set to "0":** The bit is cleared. **When set to "1":** Disabled. The state remains unchanged. **Read by read modify write instructions:** "1" is read. Note: **1)** To clear the TBOF bit, disable interrupts (TBIE = 0) or mask interrupts using the interrupt mask register (ILM) in the processor status. **2)** The TBOF bit is cleared at a write of "0", transition to main stop mode or to PLL stop mode, transition from subclock mode to main clock mode or to PLL clock mode, transition from main clock mode to PLL clock mode, at a write of "0" to the timebase timer counter clear bit (TBR), or at a reset.
bit10	TBR: Timebase timer counter clear bit	This bit clears all the bits in the timebase timer counter. **When set to "0" :** All the bits in the timebase timer counter are cleared to "0". The TBOF bit is also cleared. **When set to "1" :** Disabled. The state remains unchanged. **Read :** "1" is always read.
bit9 bit8	TBC1, TBC0: Interval time select bits	These bits set the cycle of the interval timer in the timebase timer counter. • The interval time of the timebase timer is set according to the setting of the TBC1 and TBC0 bits. • One of four time intervals can be selected.

11.4 Interrupt of Timebase Timer

The timebase timer generates an interrupt request (interval timer function) when the interval time bit in the timebase timer counter corresponding to the interval time set by the timebase timer control register carries (overflows).

■ Interrupt of Timebase Timer

- The timebase timer continues incrementing for as long as the main clock (with a half frequency of the oscillation clock) is inputted.
- When the interval time set by the interval time select bits in the timebase timer control register (TBTC: TBC1, TBC0) is reached, the interval time select bit corresponding to the interval time selected in the timebase timer counter carries and an overflow generates.
- When the interval time select bit overflows, the overflow interrupt request flag bit in the timebase timer control register (TBTC: TBOF) is set to "1".
- When the overflow interrupt request flag bit in the timebase timer control register is set (TBTC: TBOF = 1) with an interrupt enabled (TBTC: TBIE = 1), an interrupt request is generated.
- When the selected interval time is reached, the overflow interrupt request flag bit in the timebase timer control register (TBTC: TBOF) is set regardless of whether an interrupt is enabled or disabled (TBTC: TBIE)
- To clear the overflow interrupt request flag bit (TBTC: TBOF), disable a timebase timer interrupt at interrupt processing (TBTC: TBIE = 0) or mask a timebase timer interrupt by using the ILM bit in the processor status (PS) to write "0" to the TBOF bit.

Note:

An interrupt request is issued immediately if you enable interrupts (TBTC: TBIE = 1) with the overflow interrupt request flag bit in the timebase timer control register set (TBTC: TBOF = 1).

■ Correspondence between Timebase Timer Interrupt and EI^{2}OS

- The timebase timer does not correspond to EI^{2}OS.
- For details of the interrupt number, interrupt control register, and interrupt vector address, see "3.2 Interrupt Vector".

11.5 Explanation of Operations of Timebase Timer Functions

The timebase timer operates as an interval timer or an oscillation stabilization wait time timer. It also supplies a clock to peripherals.

■ Interval Timer Function

Interrupt generation at every interval time enables the timebase timer to be used as an interval timer.

Operating the timebase timer as an interval timer requires the settings shown in Figure 11.5-1 .

● Setting of timebase timer

Figure 11.5-1 Setting of Timebase Timer

Timebase timer control register (TBTC)

bit15	14	13	12	11	10	9	bit8
Re-served	—	—	TBIE	TBOF	TBR	TBC1	TBC0
1			◎	0	0	◎	◎

— : Undefined bit
◎ : Used bit
0 : Set to "0".
1 : Set to "1".

● Operations of the Interval Timer Functions

The timebase timer can be used as an interval timer by generating an interrupt at every set interval time.

- The timebase timer continues incrementing in synchronization with the main clock (a half frequency of the oscillation clock) while the oscillation clock is active.
- When the timebase timer counter reaches the interval time set by the interval time select bits in the timebase timer control register (TBTC: TBC1, TBC0), it causes an overflow (carrying) and the overflow interrupt request flag bit (TBTC: TBOF) is set to "1".
- When the overflow interrupt request flag bit is set (TBTC: TBOF = 1) with interrupts enabled (TBTC: TBIE = 1), an interrupt request is generated.

Note:
The interval time may be longer than the one set by clearing the timebase timer counter.

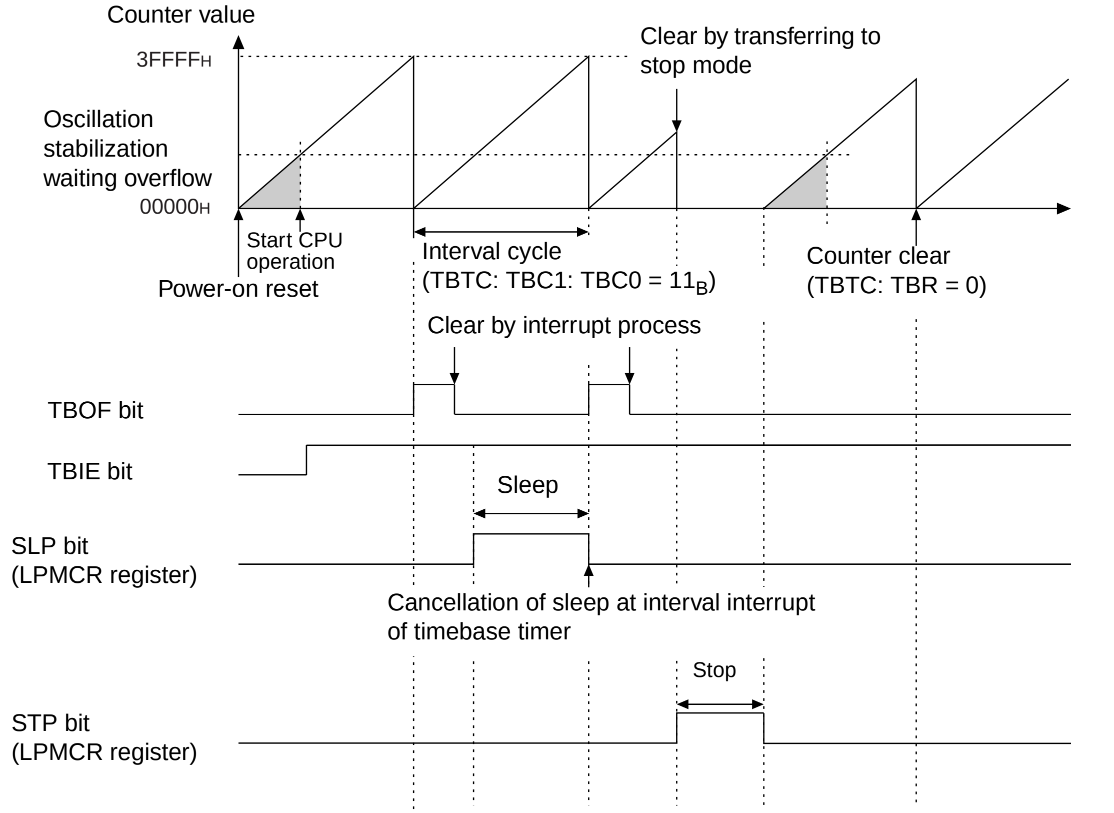
● Example of operation for timebase timer

Figure 11.5-2 gives an example of the operation that the timebase timer performs under the following conditions:

- A power-on reset occurs.
- The mode transits to the sleep mode during the operation of the interval timer.
- The mode transits to the stop mode during the operation of the interval timer.
- A request to clear the timebase timer counter is issued.

At transition to the stop mode, the timebase timer counter is cleared to stop counting up. At return from the stop mode, the timebase timer counts the oscillation stabilization wait time of the main clock.

Figure 11.5-2 Example of Operation for Timebase Timer


Counter value
3FFFFH
Clear by transferring to stop mode
Oscillation stabilization waiting overflow
00000H
Start CPU operation
Power-on reset
Interval cycle (TBTC: TBC1: TBC0 = 11B)
Counter clear (TBTC: TBR = 0)
Clear by interrupt process
TBOF bit
TBIE bit
Sleep
SLP bit (LPMCR register)
Cancellation of sleep at interval interrupt of timebase timer
Stop
STP bit (LPMCR register)

When set the interval time select bit (TBTC:TBC1, TBC0) to "11_B"(2^{19}/HCLK)

▨ : Oscillation stabilization waiting time

HCLK : Oscillation clock

■ Operation as Oscillation Stabilization Wait Time Timer

The timebase timer can be used as the oscillation stabilization wait time timer for the main clock and PLL clock.

The oscillation stabilization wait time is the time elapsed from when the timebase timer counter increments from "0" until the set oscillation stabilization wait time select bit overflows (carrying).

Table 11.5-1 shows clearing conditions and oscillation stabilization wait time of timebase timer.

Table 11.5-1 Clearing Conditions and Oscillation Stabilization Wait Time of Timebase Timer (1/2)

Operation	Counter clear	TBOF clear	Oscillation stabilization wait time
Writing "0" to timebase timer counter clear bit (TBTC: TBR)	❍	❍	
Reset			
Power-on reset	❍	❍	Transition to main clock mode after oscillation stabilization wait time of main clock completed
Watchdog reset	✕	❍	None
External reset Low voltage detection reset CPU operation detection reset Clock supervisor reset	✕	❍	None
Software reset	✕	❍	None
Switching clock mode			
Main clock → PLL clock (CKSCR: MCS=1 → 0)	❍	❍	Transition to PLL clock mode after oscillation stabilization wait time of PLL clock completed
Main clock → sub clock (CKSCR: SCS=1 → 0)	✕	✕	Transition to sub clock mode after oscillation stabilization wait time of sub clock completed
Sub clock → main clock (CKSCR: SCS=0 → 1)	❍	❍	Transition to main clock mode after oscillation stabilization wait time of main clock completed
Sub clock → PLL clock (CKSCR: MCS=0, SCS=0 → 1)	❍	❍	Transition to PLL clock mode after oscillation stabilization wait time of main clock completed
PLL clock → main clock (CKSCR: MCS=0 → 1)	✕	✕	None
PLL clock → sub clock (CKSCR: MCS=0, SCS=1 → 0)	✕	✕	None

Table 11.5-1 Clearing Conditions and Oscillation Stabilization Wait Time of Timebase Timer (2/2)

Operation	Counter clear	TBOF clear	Oscillation stabilization wait time
Cancellation of stop modes			
Cancellation of main stop mode	❍	❍	Transition to main clock mode after oscillation stabilization wait time of main clock completed
Cancellation of PLL stop mode	❍	❍	Transition to PLL clock mode after oscillation stabilization wait time of main clock completed
Cancellation of sub-stop mode	✕	✕	Transition to sub clock mode after oscillation stabilization wait time of sub clock completed
Cancellation of watch mode			
Cancellation of sub-watch mode	✕	✕	None
Cancellation of timebase timer modes			
Return to main clock mode	✕	✕	None
Return to sub clock mode	✕	✕	None
Return to PLL clock mode	✕	✕	None
Cancellation of sleep modes			
Cancellation of main sleep mode	✕	✕	None
Cancellation of sub-sleep mode	✕	✕	None
Cancellation of PLL sleep mode	✕	✕	None

■ Supply of Operation Clock

The timebase timer supplies an operation clock to the PPG timers and the watchdog timer.

Note:

Clearing the timebase timer counter may affect the operation of the resources such as the watchdog timer and PPG timers using the output of the timebase timer.

References:

- For details on the PPG timer, see "CHAPTER 16 8-/16-BIT PPG TIMER".
- For details on the watchdog timer, see "CHAPTER 12 WATCHDOG TIMER".

11.6 Precautions when Using Timebase Timer

Precautions when using the timebase timer are shown below.

■ Precautions when Using Timebase Timer

● Clearing interrupt request

To clear the overflow interrupt request flag bit in the timebase timer control register (TBTC: TBOF = 0), disable interrupts (TBTC: TBIE = 0) or mask the timebase timer interrupt by using the interrupt level mask register in the processor status.

● Clearing timebase timer counter

Clearing the timebase timer counter affects the following operations:

- When the timebase timer is used as the interval timer (interval interrupt).
- When the watchdog timer is used.
- When the clock supplied from the timebase timer is used as the operation clock of the PPG timer.

● Using timebase timer as oscillation stabilization wait time timer

After power on or in the main stop mode, PLL stop mode, and sub clock mode, the oscillation clock stops. Therefore, when oscillation starts, the timebase timer requires the oscillation stabilization wait time of the main clock. An appropriate oscillation stabilization wait time must be selected according to the types of oscillators connected to high-speed oscillation input pins.

Reference:
For details on the oscillation stabilization wait time, see "5.6 Oscillation Stabilization Wait Interval".

● Resources to which timebase timer supplies clock

- At transition to operation modes (PLL stop mode, sub clock mode, and main stop mode) in which the oscillation clock stops, the timebase timer counter is cleared and the timebase timer stops.
- When the timebase timer counter is cleared, an after-clearing interval time is needed. It may cause the clock supplied from the timebase timer to have a short High level or a 1/2 cycle longer Low level.
- The watchdog timer performs normal counting because the watchdog timer counter and timebase timer counter are cleared simultaneously.

11.7 Program Example of Timebase Timer

Programming examples for the timebase timer are shown below.

■ Program Example of Timebase Timer

- Processing specification

The 2^{12}/HCLK (HCLK: oscillation clock) interval interrupt is generated repeatedly. In this case, the interval time is approximately 1.0 ms (at 4-MHz operation).

- Coding example

```
ICR07   EQU     0000B7H         ;Timebase timer interrupt control register
TBTC    EQU     0000A9H         ;Timebase timer control register
TBOF    EQU     TBTC:3          ;Interrupt request flag bit
TBIE    EQU     TBTC:2          ;Interrupt enable bit
;-------Main program--------------------------------------
CODE    CSEG
START:                          ;Stack pointer(SP), already initialized
        AND     CCR,#0BFH       ;Interrupt disable
        MOV     I:ICR07 #00H    ;Interrupt level 0(highest)
        MOV     I:TBTC,#10000000B
                                ;Upper 3 bis are fixed
                                ;TBOF clear,
                                ;Counter clear interval time
                                ;2^12/HCLK selection
        SETB    I:TBIE          ;Interrupt enable
        MOV     ILM,#07H        ;Setting ILM in PS to level 7
        OR      CCR,#40H        ;Interrupt enable
LOOP:   MOV     A,#00H          ;No limit loop
        MOV     A,#01H
        BRA     LOOP
;-------Interrupt program------------------------------------
WARI:
        CLRB    I:TBIE          ;Clear interrupt enable bit
        CLRB    I:TBOF          ;Clear interrupt request flag
        :
      User processing
        :
        SETB    I:TBIE          ;Interrupt enable
        RETI                    ;Recovery from interrupt processing
CODE    ENDS
;-------Vector setting---------------------------------------
VECT    CSEG    ABS=0FFH
        ORG     0FF98H          ;Vector setting to interrupt number
                                 #25(19_H)
        DSL     WARI
        ORG     0FFDCH         ;Reset vector setting
        DSL     START
        DB      00H            ;Setting to single-chip mode
VECT    ENDS
        END     START
```

CHAPTER 12
WATCHDOG TIMER

This chapter describes the function and operation of the watchdog timer.

12.1 Overview of Watchdog Timer

The watchdog timer is a 2-bit counter that uses the timebase timer or watch timer as a count clock. If the counter is not cleared within a set interval time, the CPU is reset.

■ Functions of Watchdog Timer

- The watchdog timer is a timer counter that is used to prevent program malfunction. When the watchdog timer is started, the watchdog timer counter must continue to be cleared within a set interval time. If the set interval time is reached without clearing the watchdog timer counter, the CPU is reset. This is called watchdog timer.
- The interval time of the watchdog timer depends on the clock cycle input as a count clock and a watchdog reset occurs between the minimum and maximum times.
- The clock source output destination is set by the watchdog clock select bit in the watch timer control register (WTC: WDCS).
- The interval time of the watchdog timer is set by the timebase timer output select bit/watch timer output select bit in the watchdog timer control register (WDTC: WT1, WT0).

Table 12.1-1 lists the interval times of the watchdog timer.

Table 12.1-1 Interval Time of Watchdog Timer

Main

Clock cycle	Examples calculated			
	External clock(@4MHz)		CR oscillation	
	Min.	Max.	Min. (@200kHz)	Max. (@50kHz)
$2^{14} \pm 2^{11}$ /HCLK	Approx. 3.58 ms	Approx. 4.61 ms	Approx. 0.072 s	Approx. 0.369 s
$2^{16} \pm 2^{13}$ /HCLK	Approx. 14.33 ms	Approx. 18.4 ms	Approx. 0.287 s	Approx. 1.475 s
$2^{18} \pm 2^{15}$ /HCLK	Approx. 57.34 ms	Approx. 73.73 ms	Approx. 1.147 s	Approx. 5.898 s
$2^{21} \pm 2^{18}$ /HCLK	Approx. 458.75 ms	Approx. 589.82 ms	Approx. 9.175 s	Approx. 47.186 s

Sub

Clock cycle	Examples calculated			
	External clock (@32kHz, 4-frequency division)		CR oscillation	
	Min.	Max.	Min. (@200kHz)	Max. (@50kHz)
$2^{12} \pm 2^{9}$ /SCLK	Approx. 0.437 s	Approx. 0.563 s	Approx. 0.018 s	Approx. 0.092 s
$2^{15} \pm 2^{12}$ /SCLK	Approx. 3.500 s	Approx. 4.500 s	Approx. 0.143 s	Approx. 0.737 s
$2^{16} \pm 2^{13}$ /SCLK	Approx. 7.000 s	Approx. 9.000 s	Approx. 0.287 s	Approx. 1.475 s
$2^{17} \pm 2^{14}$ /SCLK	Approx. 14.000 s	Approx. 18.000 s	Approx. 0.573 s	Approx. 2.949 s

Note: See "CHAPTER 6 CLOCK SUPERVISOR" for the CR oscillation.

Notes:

- When the timebase timer output (carry signal) is used as a count clock to the watchdog timer, clearing the timebase timer may extend the time for a watchdog reset to occur.
- When the subclock is used as the machine cock, be sure to set the watchdog timer clock source select bit (WDCS) in the watch timer control register (WTC) to "0" to select the watch timer output.

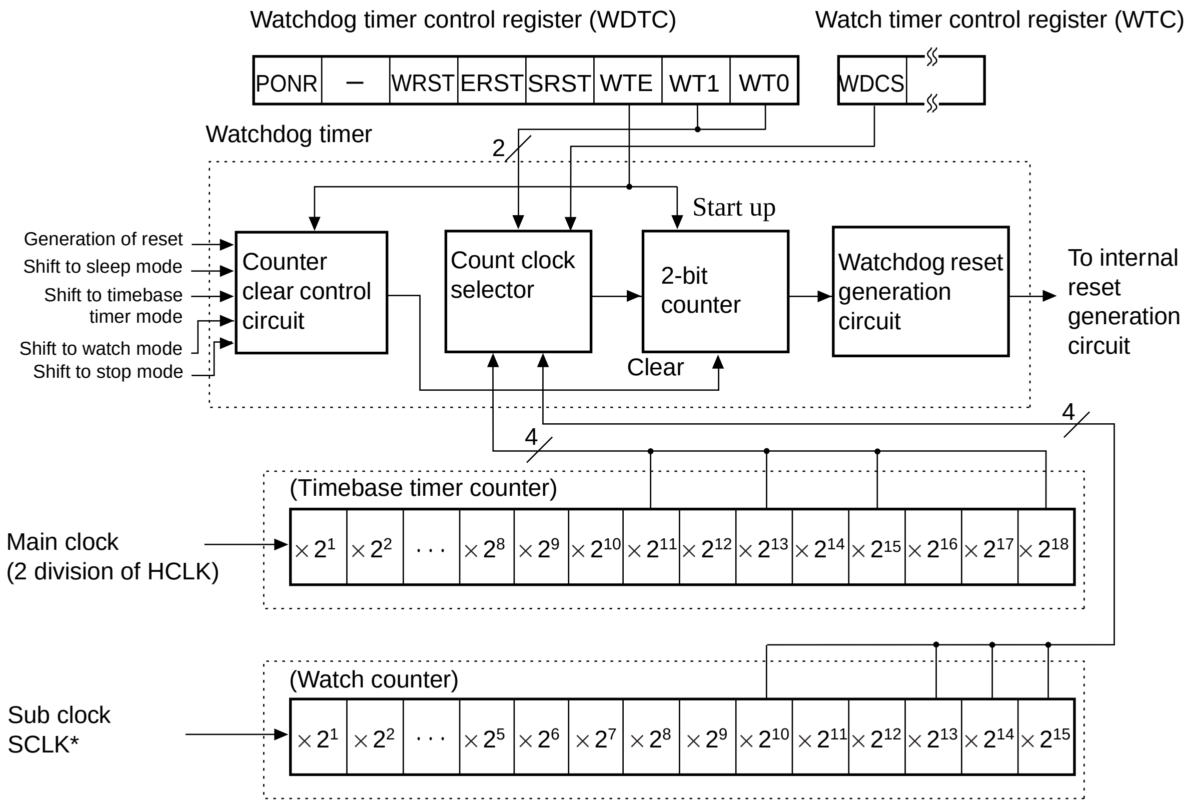
12.2 Configuration of Watchdog Timer

The watchdog timer consists of the following blocks:

- **Count clock selector**
- **Watchdog timer counter (2-bit counter)**
- **Watchdog reset generator**
- **Counter clear control circuit**
- **Watchdog timer control register (WDTC)**

■ Block Diagram of Watchdog Timer

Figure 12.2-1 Block Diagram of Watchdog Timer



HCLK : Oscillation clock
SCLK : Sub clock
* : SCLK is 2 division or 4 division of the clock inputted to the low-speed oscillation pin (X0A and X1A) or internal CR oscillation clock. The division ratio is set by the SCDS bit of the PLL/subclock control register (PSCCR). (See "CHAPTER 5 CLOCKS".)

- Count clock selector

The count clock selector selects a count clock input to the watchdog timer from the timebase timer or watch timer. Each timer output has four time intervals that can be set.

- Watchdog timer counter (2-bit counter)

The watchdog timer counter is a 2-bit counter that uses the timebase timer output or watch timer output as a count clock. The clock source output destination is set by the watchdog clock select bit in the watch timer control register (WTC: WDCS).

- Watchdog reset generator

The watchdog reset generation circuit generates a reset signal when the watchdog timer overflows (carrying).

- Counter clear circuit

The counter clear circuit clears the watchdog timer counter.

- Watchdog timer control register (WDTC)

The watchdog timer control register starts and clears the watchdog timer, sets the interval time, and holds reset factors.

12.3 Watchdog Timer Registers

This section explains the registers used for setting the watchdog timer.

■ List of Registers and Reset Values of Watchdog Timer

Figure 12.3-1 List of Registers and Reset Values of Watchdog Timer

		bit 7	6	5	4	3	2	1	0
Watchdog timer control register (WDTC)	Address 0000A8$_H$	×	×	×	×	×	1	1	1

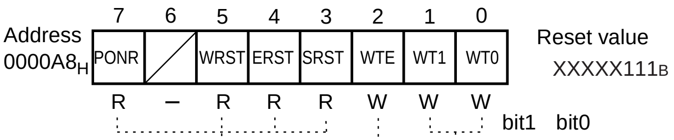
× : Undefined

12.3.1 Watchdog timer control register (WDTC)

The watchdog timer control register starts and clears the watchdog timer, sets the interval time, and holds reset factors.

■ Watchdog Timer Control Register (WDTC)

Figure 12.3-2 Watchdog Timer Control Register (WDTC)



bit1 bit0

WT1	WT0	Interval time select bit (timebase timer output select)		
		Interval time		Clock cycle
		Min	Max	
0	0	approx. 3.58 ms	approx. 4.61 ms	$2^{14} \pm 2^{11}$/HCLK
0	1	approx. 14.33 ms	approx. 18.3 ms	$2^{16} \pm 2^{13}$/HCLK
1	0	approx. 57.23 ms	approx. 73.73 ms	$2^{18} \pm 2^{15}$/HCLK
1	1	approx. 458.75 ms	approx. 589.82 ms	$2^{21} \pm 2^{18}$/HCLK

HCLK: Oscillation clock
The parenthesized values are interval time when the oscillation clock operates at HCLK 4 MHz.

bit1 bit0

WT1	WT0	Interval time select bit (watch timer output select)		
		Interval time		Clock cycle
		Min	Max	
0	0	approx. 0.457 s	approx. 0.576 s	$2^{12} \pm 2^{9}$/SCLK
0	1	approx. 3.584 s	approx. 4.608 s	$2^{15} \pm 2^{12}$/SCLK
1	0	approx. 7.168 s	approx. 9.216 s	$2^{16} \pm 2^{13}$/SCLK
1	1	approx. 14.336 s	approx. 18.432 s	$2^{17} \pm 2^{14}$/SCLK

SCLK: Sub clock*2
The parenthesized values are interval time when the oscillation clock

bit2

WTE	Watchdog timer control bit	
0	First programming after reset: Start up the watchdog timer	Twice or more programming after reset : Clear the watchdog timer
1	No effect	

bit7 bit5 bit4 bit3

Reset factor bit				Reset factor
PONR	WRST	ERST	SRST	
1	X	X	X	Power-on reset
*1	1	*1	*1	Watchdog reset
*1	*1	1	*1	External reset (Low level input to $\overline{\text{RST}}$ pin)
*1	*1	*1	1	Software reset (write "1" to RST bit)

R : Read only
W : Write only
X : Undefined

*1 : The previous state is held.

*2 : However, SCLK is 2 division or 4 division of the clock inputted to the low-speed oscillation pin (X0A and X1A) or internal CR oscillation clock. The division ratio is set by the SCDS bit of the PLL/subclock control register (PSCCR). (See "CHAPTER 5 CLOCKS".)
See Table 12.1-1 for the interval time.

Table 12.3-1 Functions of the Watching Timer Control Register (WDTC)

Bit name		Function
bit0 bit1	WT1, WT0: Interval time select bits	These bits set the interval time of the watchdog timer. The time interval when the watch timer is used as the clock source to the watchdog timer (watchdog clock select bit WDCS= 0) is different from when the main clock mode or the PLL clock mode is selected as the clock mode and the WDCS bit in the watch timer control register (WTC) is set to "1" as shown in Figure 12.3-2 according to the settings of the WTC register. In the subclock mode, be sure to set the watchdog clock select bit (WDCS) in the watch timer control register (WTC) to "0" and select the output of the watch timer. • Data after the watchdog timer is started is valid only. • Write data after the watchdog timer is started is ignored. • These are write-only bits.
bit2	WTE: Watchdog timer control bit	This bit starts or clears the watchdog timer. **When set to "0" (first time after reset):** The watchdog timer is started. **When set to "0" (second or subsequent):** The watchdog timer is cleared.
bit6	Undefined bit	**Read:** The value is undefined. **Write:** No effect
bit3 to bit5, bit7	PONR, WRST, ERST, SRST: Reset factor bits	These bits indicate reset factors. • When a reset occurs, the bit corresponding to the reset factor is set to "1". After a reset, the reset factor can be checked by reading the watchdog timer control register (WDTC). • These bits are cleared after the watchdog timer control register (WDTC) is read. Note: No bit value other than the PONR bit after power-on reset is assured. If the PONR bit is set at read, other bit values should be ignored.

12.4 Explanation of Operations of Watchdog Timer Functions

After starting, when the watchdog timer reaches the set interval time without the counter being cleared, a watchdog reset occurs.

■ Operations of Watchdog Timer

The operation of the watchdog timer requires the settings shown in Figure 12.4-1 .

Figure 12.4-1 Setting of Watchdog Timer

	bit7	6	5	4	3	2	1	bit0
Watchdog timer control register (WDTC)	PONR	–	WRST	ERST	SRST	WTE	WT1	WT0
						0	❍	❍

	bit7	6	5	4	3	2	1	bit0
Watch timer control register (WTC)	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0
	❍							

❍ : Used bit
0 : Set to "0".

● Selecting clock input source

- The timebase timer or watch timer can be selected as the clock input source of the count clock to the watchdog timer. When the watchdog clock select bit (WTC: WDCS) is set to "1", the timebase timer is selected. When the bit is set to 0, the watch timer is selected. After a reset, the bit returns to "1".
- During operation in the sub clock mode, set the WDCS bit to 0 to select the watch timer.

● Setting interval time

- Set the interval time select bits (WDTS: WT1, WT0) to select the interval time for the watchdog timer.
- Set the interval time concurrently when starting the watchdog timer. Writing to the bit is ignored after the watchdog timer is started.

● Activating watchdog timer

When "0" is written to the watchdog timer control bit (WDTC: WTE) after a reset, the watchdog timer is started and starts incrementing.

● Clearing watchdog timer

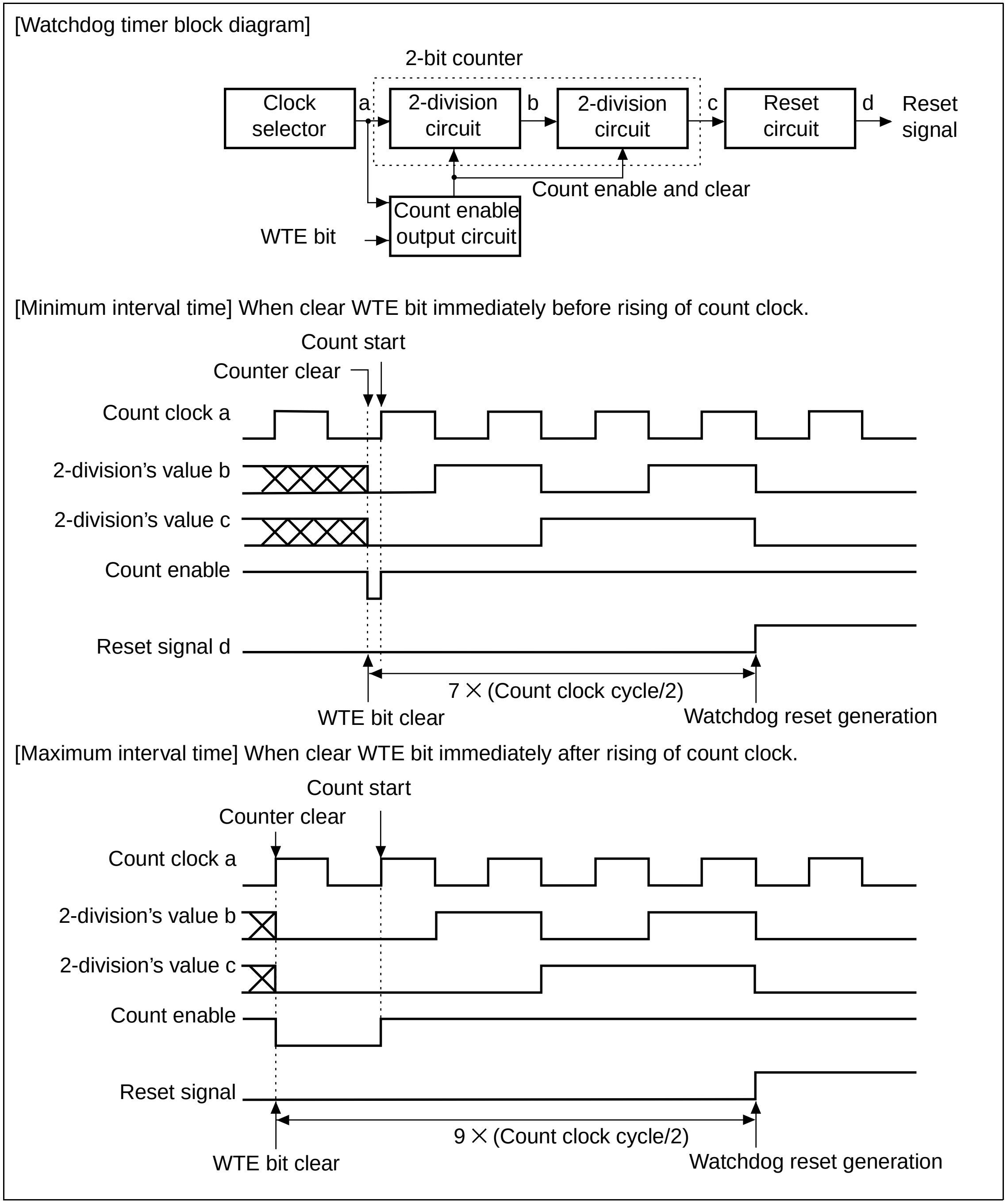
- When "0" is written once again to the watchdog timer control bit (WDTC: WTE) within the interval time after starting the watchdog timer, the watchdog timer is cleared. If the watchdog timer is not cleared within the interval time, it overflows and the CPU is reset.
- A reset, or transitions to the standby modes (sleep mode, stop mode, watch mode, timebase timer mode) clear the watchdog timer.
- During operation in the timebase timer mode or watch mode, the watchdog timer counter is cleared. However, the watchdog timer remains in the activation state.
- Figure 12.4-2 shows relationship between clear timing and interval time of watchdog timer. The interval time varies with the timing of clearing the watchdog timer.

● Checking reset factors

The reset factor bits in the watchdog timer control register (WDTC: PONR, WRST, ERST, SRST) can be read after a reset to check the reset factors.

Reference:	For details on the reset factor bit, see "CHAPTER 7 RESETS".

Figure 12.4-2 Relationship between Clear Timing and Interval Time of Watchdog Timer



12.5 Precautions when Using Watchdog Timer

Take the following precautions when using the watchdog timer.

■ Precautions when Using Watchdog Timer

● Stopping watchdog timer

The watchdog timer is stopped by all the reset sources.

● Interval time

- The interval time uses the carry signal of the timebase timer or watch timer as a count clock. If the timebase timer or watch timer is cleared, the interval time of the watchdog timer may become long. Note that the timebase timer is cleared when "0" is written to the timebase timer counter clear bit (TBR) in the timebase timer control register (TBTC) and when the clock mode changes from the main clock to PLL clock, from the subclock to main clock, or from the subclock to PLL clock.
- Set the interval time concurrently when starting the watchdog timer. Setting the time interval except starting the watchdog timer is ignored.

● Precautions when creating program

When clearing the watchdog timer repeatedly in the main loop, set a shorter processing time for the main loop, including interrupt processing, than the interval time of watchdog timer.

● Precautions in subclock mode

In the subclock mode, be sure to set the watchdog clock select bit (WDCS) in the watch timer control register (WTC) to "0" and select the output of the watch timer.

12.6 Program Examples of Watchdog Timer

Program example of watchdog timer is given below:

■ Program Examples of Watchdog Timer

● Processing specification

- The watchdog timer is cleared each time in the loop of the main program.
- The main program must be executed once within the minimum interval time of the watchdog timer.

● Coding example

```
WDTC    EQU    0000A8H            ;Watchdog timer control
                                   register
WTE     EQU    WDTC:2             ;Watchdog control bit
;
;---------Main program-------------------------------------
CODE           CSEG
START:                            ;Stack pointer (SP), already
                                  ;initialized
        MOV    I:WDTC,#00000011B  ;Start up of watchdog timer
                                  ;Select interval time $2^{21}$+$2^{18}$
                                  ;cycle
LOOP:
        CLRB   I:WTE              ;Clear watchdog timer
        ÅE
        User processing
        ÅE
        BRA    LOOP
;---------Vector setting-------------------------------------------
VECT    CSEG   ABS=0FFH
        ORG    00FFDCH            ;Reset vector setting
        DSL    START
        DB     00H                ;Setting to single chip mode
VECT    ENDS
        END    START
```

CHAPTER 13

16-Bit I/O TIMER

This chapter explains the function and operation of the 16- bit I/O timer.

13.1 Overview of 16-bit I/O Timer

The 16-bit I/O timer consists of one 16-bit free-run timer and 4 input capture. The timer can be performed the measurement of input pulse and external clock cycle based on the 16-bit free-run timer.

■ Module Configuration of 16-bit I/O Timer

The 16-bit I/O timer consists of the following modules:

- 16-bit free-run timer × 1 unit

  16-bit free-run timer 0 (channel 0)

- Input capture × 4 units

  Input capture unit 0: capture 16-bit free-run timer 0

  - Input capture 0 (channel 0)
  - Input capture 1 (channel 1)
  - Input capture 2 (channel 2)
  - Input capture 3 (channel 3)

■ Functions of 16-bit I/O Timer

● Functions of 16-bit free-run timer

The 16-bit free-run timer consists of a 16-bit up counter, a prescaler, and a control register.

The count value of the 16-bit free-run timer can be use as the base time for the input capture.

- One of eight types of the count clock cycle can be set.
- An overflow in the counter generates an interrupt request.
- The counter of the 16-bit free-run timer is cleared to "0000_H" by reset or timer clear (TCCSL:CLR=1).

● Functions of input capture

The input capture consists of four 16-bit capture registers and control registers corresponding to the external input pin, and the edge detection circuit.

When the trigger edge is inputted to the external input pin, the counter value of the 16-bit free-run timer is retained and the interrupt request is generated at the same time.

- The capture interrupt can be generated independently by each channel.
- The EI^2OS can be started.
- Trigger edge can be selected from rising edge, falling edge, or both edges.
- Because each channel operates independently, up to 4 input measurement is performed.
- When the input signal is set to the LIN-UART, the baud rate measurement at LIN slave operation is executed.

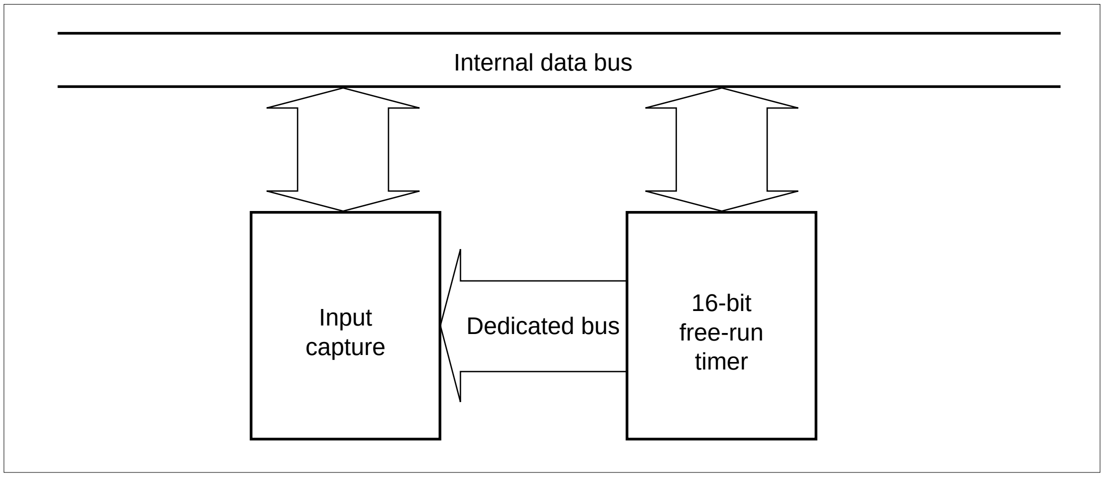
13.2 Block Diagram of 16-bit I/O Timer

The 16-bit I/O timer consists of the following modules:

- **16-bit free-run timer**
- **Input capture**

■ Block Diagram of 16-bit I/O Timer

Figure 13.2-1 Block Diagram of 16-bit I/O Timer



● 16-bit free-run timer

The count value of the 16-bit free-run timer can be used as the base time for the input capture.

● Input capture

When the trigger edge is inputted to the external input pin, or when the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted, the counter value of the 16-bit free-run timer is retained and the interrupt request is generated at the same time.

■ Details of Pins and Interrupt Number

Table 13.2-1 shows the pins used by the 16-bit details of interrupt.

Table 13.2-1 Details of Pins and Interrupt Number

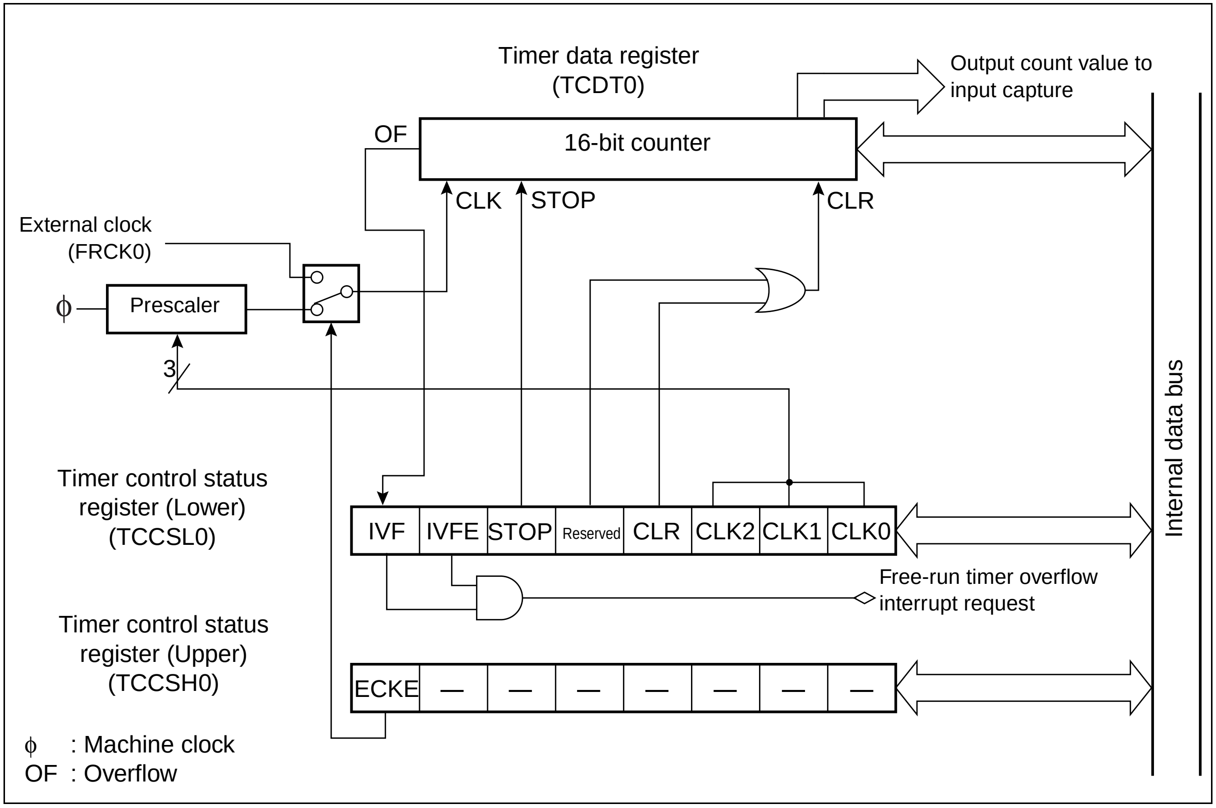
Channel	Special terminal	Pin name	Interrupt No.	For I^2OS
Input capture ch0 (using 16-bit free-run timer ch0)	IN0	P24/IN0	#33 (21_H)	❍
Input capture ch1 (using 16-bit free-run timer ch0)	IN1	P25/IN1		
Input capture ch2 (using 16-bit free-run timer ch0)	IN2	P26/IN2		
Input capture ch3 (using 16-bit free-run timer ch0)	IN3	P27/IN3		
16-bit free-run timer ch0 (overflow interrupt)	FRCK0	P44/FRCK0	#30 ($1E_H$)	×

13.2.1 Block Diagram of 16-bit Free-run Timer

The MB90360 series contains 1 channel of the 16-bit free-run timer, and it consists of the following block.

■ Block Diagram of 16-bit Free-run Timer

Figure 13.2-2 Block Diagram of 16-bit Free-run Timer



- Prescaler

The prescaler divides the frequency of the machine clock to supply a count clock to the 16-bit counter. Any of eight count clock cycles can be selected by setting the timer control status register (TCCSL: CLK2 to CLK0).

- Timer data register (TCDT)

The timer data register can read the counter value of the 16-bit free-run timer. During stopping of the 16-bit free-run timer, the counter value can be set by writing the counter value to the TCDT.

- Timer control status register (TCCSH, TCCSL)

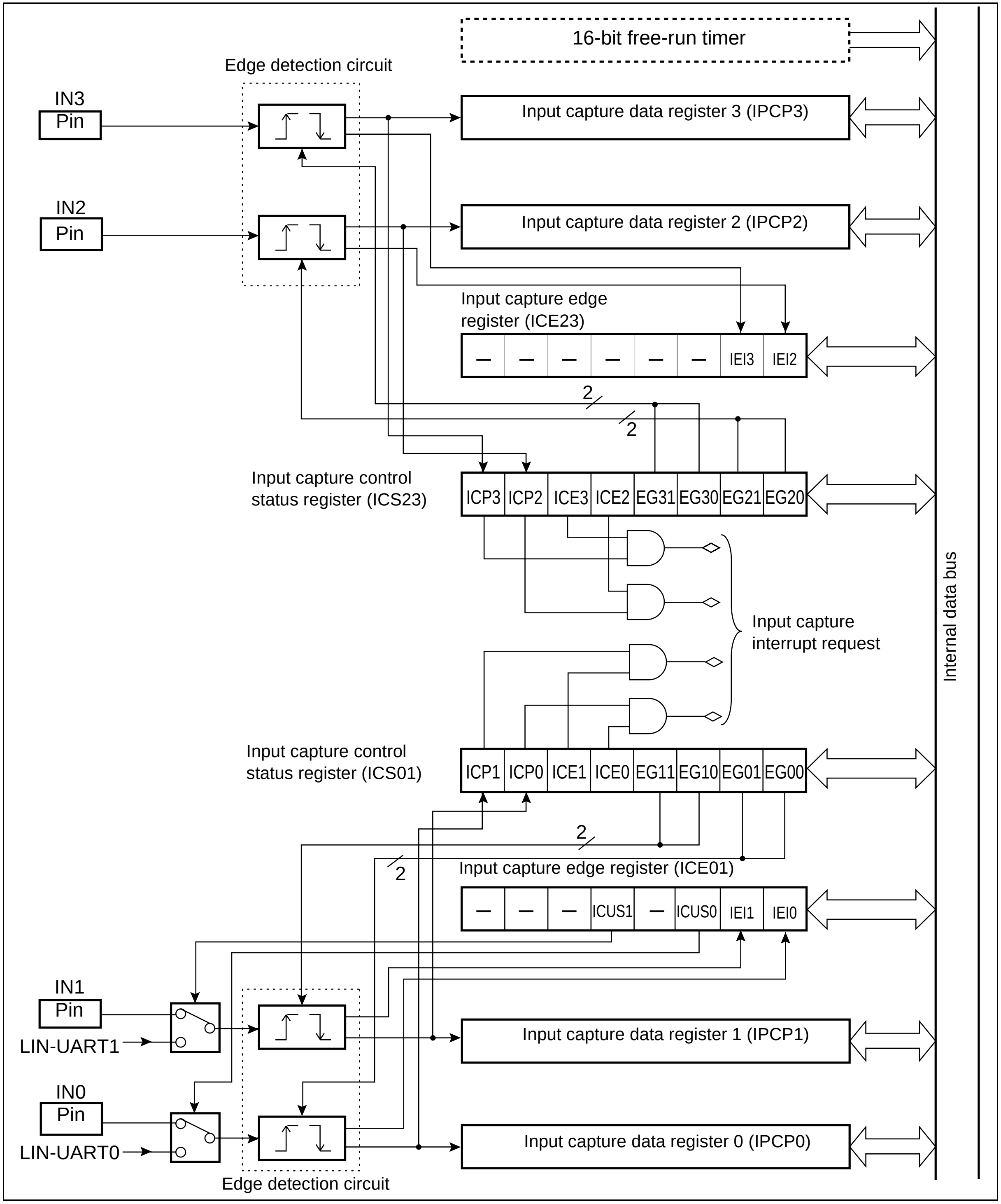
The timer control status register (upper and lower) selects the count clock and the condition for clearing the counter, clears the counter, enables the count operation and interrupt request, checks the overflow generation flag.

13.2.2 Block Diagram of Input Capture

The input capture consist of the following blocks:

■ Block Diagram of Input Capture

Figure 13.2-3 Block Diagram of Input Capture Unit 0


16-bit free-run timer
Edge detection circuit
IN3
Pin
Input capture data register 3 (IPCP3)
IN2
Pin
Input capture data register 2 (IPCP2)
Input capture edge register (ICE23)
—
—
—
—
—
—
IEI3
IEI2
2
2
Input capture control status register (ICS23)
ICP3
ICP2
ICE3
ICE2
EG31
EG30
EG21
EG20
Input capture interrupt request
Internal data bus
Input capture control status register (ICS01)
ICP1
ICP0
ICE1
ICE0
EG11
EG10
EG01
EG00
2
2
Input capture edge register (ICE01)
—
—
—
ICUS1
—
ICUS0
IEI1
IEI0
IN1
Pin
LIN-UART1
Input capture data register 1 (IPCP1)
IN0
Pin
LIN-UART0
Input capture data register 0 (IPCP0)
Edge detection circuit

● Input capture data registers 0 to 3 (IPCP0 to IPCP3)

- Input capture data register retains the counter value of the 16-bit free-run timer fetched by the capture operation.
- Input capture data register 0 to 3 keep the counter value of the 16-bit free-run timer 0

● Input capture control status registers 01 to 23 (ICS01 to ICS23)

- Input capture control status register selects the trigger edge, enables the capture operation and capture interrupt request, and checks the valid edge detection flag for each input capture.
- Input capture control status register has 2 registers, and the input capture operation of the corresponding channel is controlled as shown in Table 13.2-2 .

● Input capture edge registers 01 to 23 (ICS01 to ICS23)

- Input capture control status register indicated the edge polarity detected by each input capture. Also, it selects the input signal (external pin INx/LIN-UART). When input is set to the LIN-UART, the baud rate measurement at the LIN slave operation can be performed (See "20.7.3 Operation with LIN Function (Operation Mode 3)").
- Input capture edge register has 2 registers, and the input capture operation of the corresponding channel is controlled as shown in Table 13.2-2 .

Table 13.2-2 Relationship between the Register and Pin of Input Capture

	Input capture control status register	Input capture edge register	Input capture data register	Input pin	Input form LIN-UART
Input capture unit 0	ICS01	ICE01	IPCP0	IN0	UART0
			IPCP1	IN1	UART1
	ICS23	ICE23	IPCP2	IN2	-
			IPCP3	IN3	-

● Edge detection circuit

The edge detection circuit detects the edge of the signal input to the external input pin. The detected edge can be selected from among the rising edge, falling edge, both edges, and no detection (capture stop).

13.3 Configuration of 16-bit I/O Timer

This section explains the pins, registers, and interrupt factors of the 16-bit I/O timer.

■ Pins of 16-bit I/O Timer

The pins of the 16-bit I/O timer serve as general-purpose I/O ports. Table 13.3-1 shows the pin functions and the pin settings required to use the 16-bit I/O timer.

Table 13.3-1 Pins of 16-bit I/O Timer

Channel	Pin Name	Pin Function	Setting to use the pin
16-bit free-run timer 0	P44/ FRCK0	General-purpose I/O port, external clock input	Set as input port in port direction register (DDR).
Input capture 0	P24/IN0	General-purpose I/O port, capture input	Set as input port in port direction register (DDR).
Input capture 1	P25/IN1		Set as input port in port direction register (DDR).
Input capture 2	P26/IN2		Set as input port in port direction register (DDR).
Input capture 3	P27/IN3		Set as input port in port direction register (DDR).

■ Generation of Interrupt Request from 16-bit I/O Timer

The 16-bit I/O timer can generate an interrupt request as a result of the following factors:

● Timer counter overflow interrupt

If the overflow interrupt request is set to enable (TCCSL: IVFE=1), the interrupt is occurred by the following factor:

- 16-bit free-run timer overflow

● Input capture interrupt

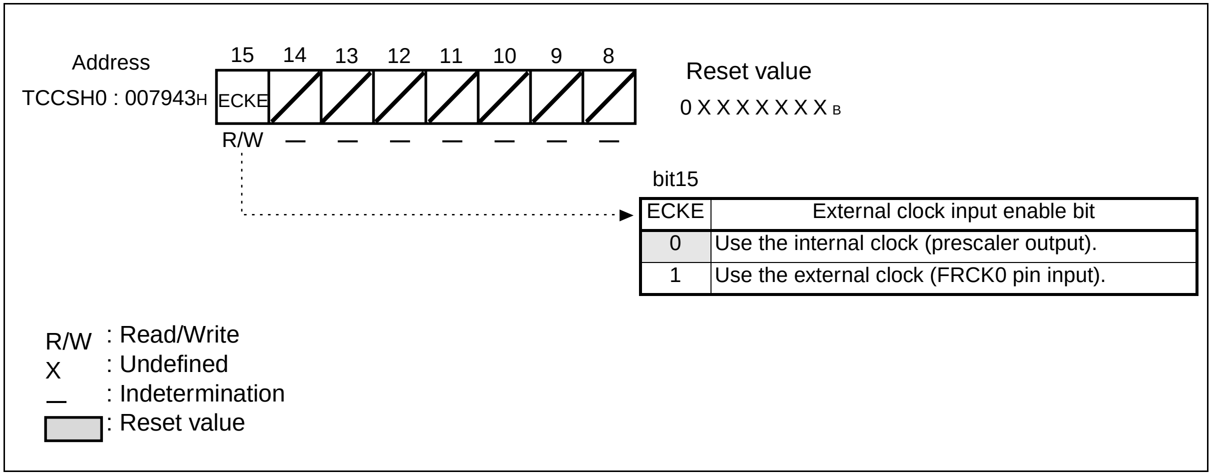
If the input capture interrupt request is set to enable (ICS: ICE=1), if the trigger edge is detected by the input capture pin, or if the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted, the interrupt request is generated.

13.3.1 Timer Control Status Register (Upper) (TCCSH)

Timer control status register (upper) selects the count clock and the conditions for clearing the counter, enables the count operation and interrupt, and checks the interrupt request flag.

■ Timer Control Status Register (Upper) (TCCSH)

Figure 13.3-1 Timer Control Status Register (Upper) (TCCSH)



ECKE	External clock input enable bit
0	Use the internal clock (prescaler output).
1	Use the external clock (FRCK0 pin input).

Table 13.3-2 Function of Timer Control Status Register (Upper) (TCCSH)

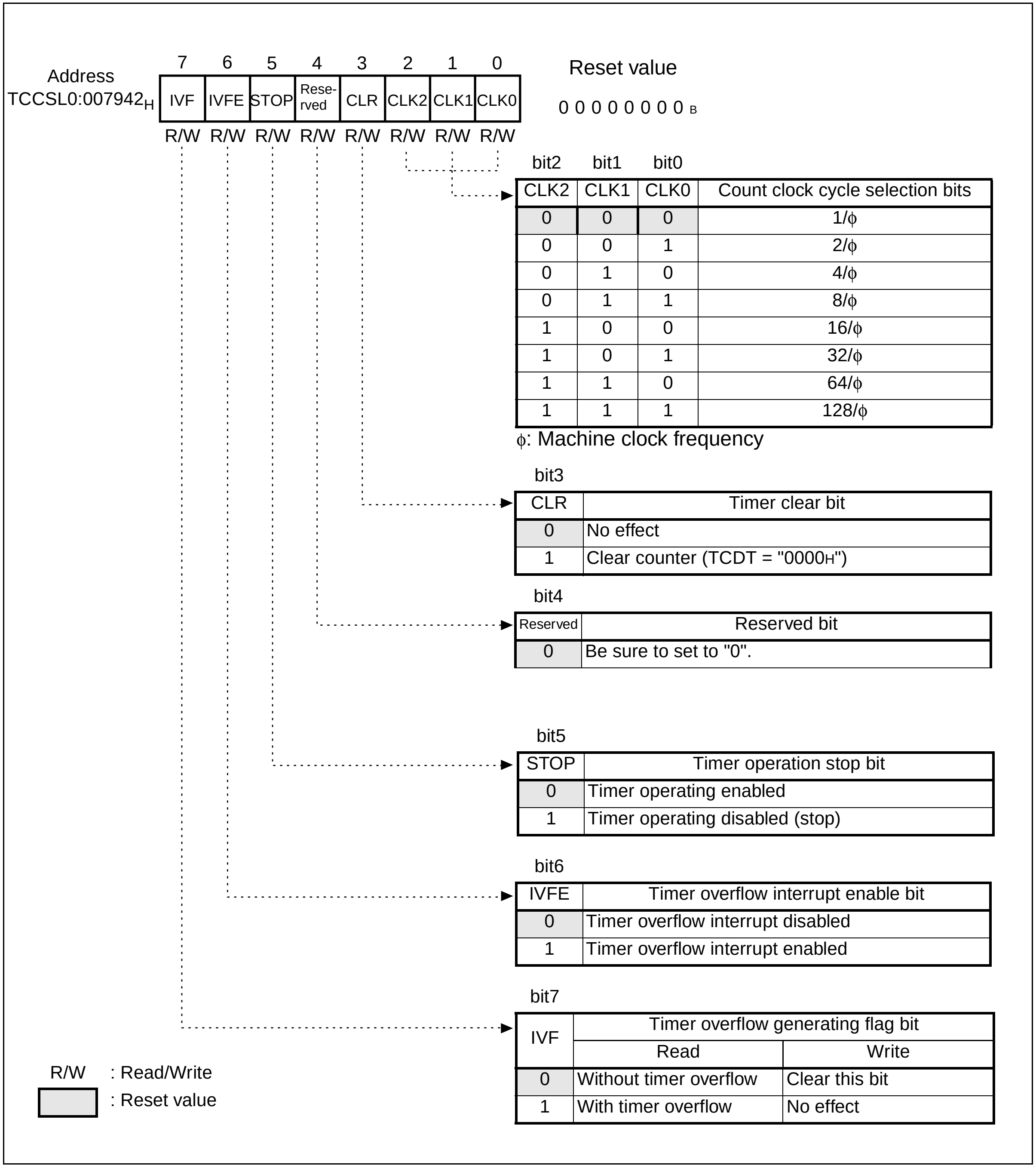
Bit name		Function
bit15	ECKE : External clock input enable bit	This bit selects the count clock of the 16-bit free-run timer. **When set to "1":** Use the clock inputted from the external pin FRCK0. **When set to "0":** Use the internal clock (clock outputted from the prescaler). Note: Set the ECKE bit during stopping of the free-run timer (TCCSL:STOP=1).
bit14 to bit8	Undefined bits	**Read:** The value is undefined **Write:** No effect

13.3.2 Timer Control Status Register (Lower) (TCCSL)

The timer control status register (Lower) selects the count clock and conditions for clearing the counter, clears the counter, enables the count operation or interrupt, and checks the interrupt request flag.

■ Timer Control Status Register (Lower) (TCCSL)

Figure 13.3-2 Timer Control Status Register (Lower) (TCCSL)



Address	7	6	5	4	3	2	1	0	Reset value
TCCSL0:007942H	IVF	IVFE	STOP	Rese-rved	CLR	CLK2	CLK1	CLK0	0 0 0 0 0 0 0 0 B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

bit2 bit1 bit0

CLK2	CLK1	CLK0	Count clock cycle selection bits
0	0	0	1/φ
0	0	1	2/φ
0	1	0	4/φ
0	1	1	8/φ
1	0	0	16/φ
1	0	1	32/φ
1	1	0	64/φ
1	1	1	128/φ

φ: Machine clock frequency

bit3

CLR	Timer clear bit
0	No effect
1	Clear counter (TCDT = "0000H")

bit4

Reserved	Reserved bit
0	Be sure to set to "0".

bit5

STOP	Timer operation stop bit
0	Timer operating enabled
1	Timer operating disabled (stop)

bit6

IVFE	Timer overflow interrupt enable bit
0	Timer overflow interrupt disabled
1	Timer overflow interrupt enabled

bit7

IVF	Timer overflow generating flag bit	
	Read	Write
0	Without timer overflow	Clear this bit
1	With timer overflow	No effect

R/W : Read/Write
▨ : Reset value

Table 13.3-3 Functions of Timer Control Status Register (Lower) (TCCSL)

Bit name		Function
bit7	IVF: Timer overflow generation flag bit	This bit indicates the timer overflow. **[Condition set to "1"]** Condition is set when the following is used. • When 16-bit free-run timer overflows **[When set to "1"]** When the timer overflow interrupt request is set to enable (TCCSL:IVFE=1) if the IVF bit is set to "1", the interrupt request is generated. **When set to "0": The bit is cleared.** **When set to "1": No effect.** **Read by read modify write instructions: "1" is always read.**
bit6	IVFE: Timer overflow interrupt enable bit	This bit enables or disables the interrupt request when the IVF bit is set to "1". **When set to "1":** When the IVF bit is set to "1", the interrupt request is generated. **When set to "0":** The generation of the interrupt request is disabled.
bit5	STOP: Timer operation stop bit	This bit enables or disables (stops) the operation of the 16-bit free-run timer. **When set to "0":** Enable the timer operation and count up with count clock set by the CLK2 to CLK0. **When set to "1":**Stops count operation
bit4	Reserved bit	Always set this bit to "0".
bit3	CLR: Timer clear bit	This bit clears the counter (TCDT) of the 16-bit free-run timer. **When set to "1": Clears timer data register (TCDT) to "0000$_{H}$"** **When set to "0": No effect.** **Read: "0" is always read.** Note: When clearing during stopping of the 16-bit free-run timer (TCCSL:STOP=1), write "0000$_{H}$" to the TCDT directly.
bit2 bit1 bit0	CLK2, CLK1, CLK0: Count clock cycle selection bits	These bits set the count clock to cycle of the 16-bit free-run timer. Note: Set the count clock cycle during stopping of the input capture operation (ICSnm: EGn1, EGn0="00$_{B}$" or ICSnm:EGm1, EGm0="00$_{B}$").

n=0, 2 m=n+1

13.3.3 Timer Data Register (TCDT)

The timer data register is a 16-bit up counter.

- **The counter value of the 16-bit free-run timer is read.**
- **The counter value can be set during stopping of the 16-bit free-run timer.**

■ Timer Data Register (TCDT)

Figure 13.3-3 Timer Data Register (TCDT)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Reset value
TCDT0 upper: 007941_H	Tn15	Tn14	Tn13	Tn12	Tn11	Tn10	Tn9	Tn8	00000000_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset value
TCDT0 lower: 007940_H	Tn7	Tn6	Tn5	Tn4	Tn3	Tn2	Tn1	Tn0	00000000_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Read/Write

n=0

The TCDT register can read the counter value of the 16-bit free-run timer.

[Condition for clear the counter value]

The counter value is cleared to "0000_H" by the following conditions.

- Overflow
- Setting of "1" to the timer clear bit of the timer control status register (TCCSL:CLR=1)
- Setting of "0000_H" to the timer data register during stopping of 16-bit free-run timer
- Reset

[Setting of counter value]

Write the counter value to the timer data register (TCDT) and set the timer during stopping the timer operation (TCCSL:STOP=1).

Note:

Always use a word instruction (MOVW) to read/write the timer data register.

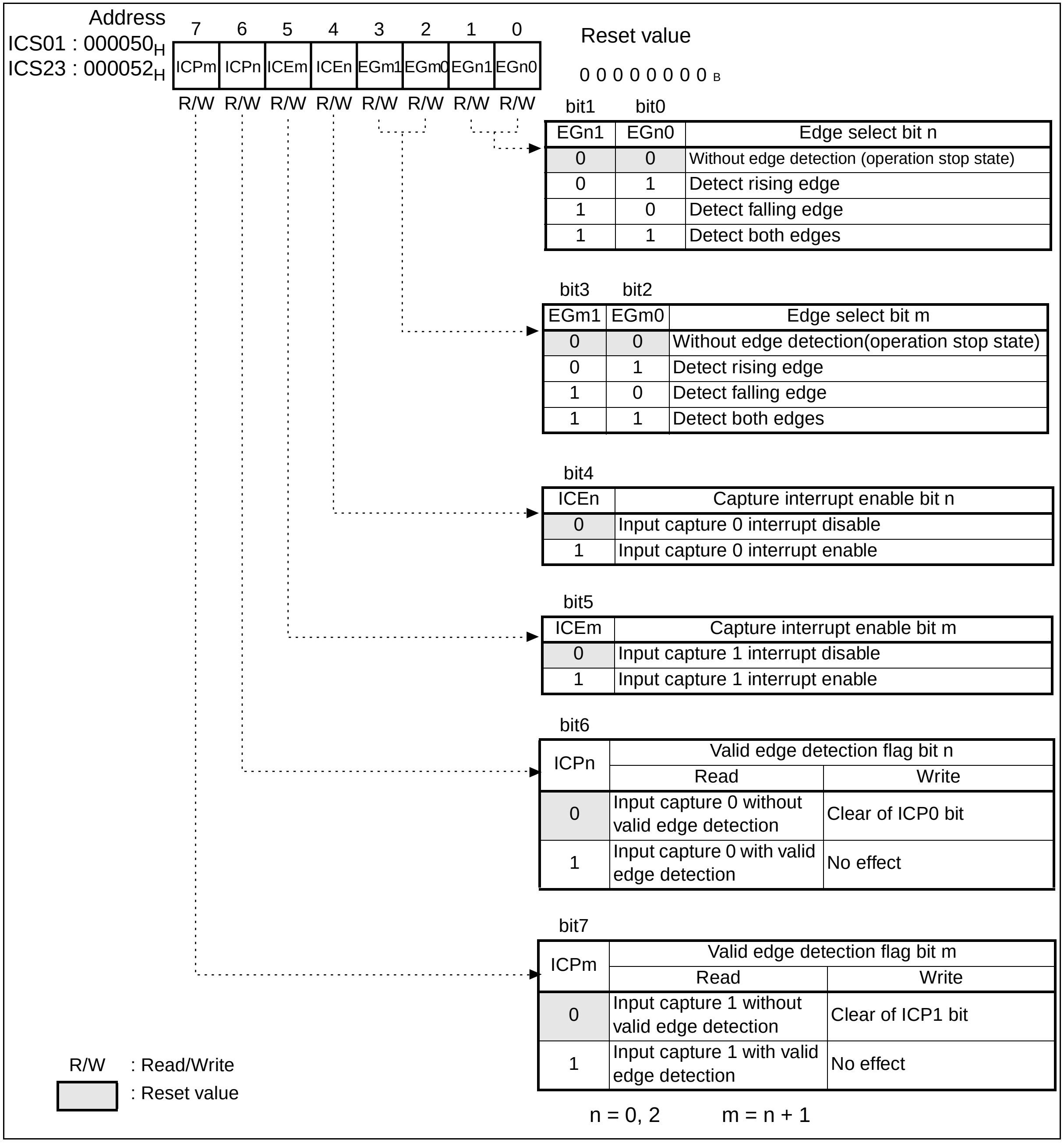
13.3.4 Input Capture Control Status Registers (ICS)

The function of the input capture control status register is shown below.
The correspondence between ICS01 to ICS23 and input pin is as follows.

- **ICS01: IN0, IN1 input capture ch0, ch1**
- **ICS23: IN2, IN3 input capture ch2, ch3**

■ Input Capture Control Status Registers (ICS01, ICS23)

Figure 13.3-4 Input Capture Control Status Registers (ICS)



Address
ICS01 : 000050_H
ICS23 : 000052_H

7	6	5	4	3	2	1	0
ICPm	ICPn	ICEm	ICEn	EGm1	EGm0	EGn1	EGn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Reset value
0 0 0 0 0 0 0 0 B

bit1	bit0	
EGn1	EGn0	Edge select bit n
0	0	Without edge detection (operation stop state)
0	1	Detect rising edge
1	0	Detect falling edge
1	1	Detect both edges

bit3	bit2	
EGm1	EGm0	Edge select bit m
0	0	Without edge detection(operation stop state)
0	1	Detect rising edge
1	0	Detect falling edge
1	1	Detect both edges

bit4	
ICEn	Capture interrupt enable bit n
0	Input capture 0 interrupt disable
1	Input capture 0 interrupt enable

bit5	
ICEm	Capture interrupt enable bit m
0	Input capture 1 interrupt disable
1	Input capture 1 interrupt enable

bit6

ICPn	Valid edge detection flag bit n	
	Read	Write
0	Input capture 0 without valid edge detection	Clear of ICP0 bit
1	Input capture 0 with valid edge detection	No effect

bit7

ICPm	Valid edge detection flag bit m	
	Read	Write
0	Input capture 1 without valid edge detection	Clear of ICP1 bit
1	Input capture 1 with valid edge detection	No effect

R/W : Read/Write
▨ : Reset value

n = 0, 2 m = n + 1

Table 13.3-4 Functions of Input Capture Control Status Register (ICS)

Bit name		Function
bit7	ICPm: Valid edge detection flag bit m	This bit is set to "1" when the valid edge is detected by the INm pin. When the interrupt request of the input capture m is set to enable (ICSnm:ICEm=1), if the ICPm bit is set, the interrupt request is generated. **When set to "0": The bit is cleared.** **When set to "1": No effect.**
bit6	ICPn: Valid edge detection flag bit n	This bit is set to "1" when the valid edge is detected by the INn pin. When the interrupt request of the input capture m is set to enable (ICSnm:ICEn=1), if the ICPn bit is set, the interrupt request is generated. **When set to "0": The bit is cleared.** **When set to "1": No effect.**
bit5	ICEm: Capture interrupt enable bit m	This bit enables or disables the interrupt request of the input capture m. **When set to "1":** When the valid edge detection flag bit m is set to "1" (ICSnm: ICPm=1), the interrupt request is generated.
bit4	ICEn: Capture interrupt enable bit n	This bit enables or disables the interrupt request of the input capture n. **When set to "1":** When the valid edge detection flag bit n is set to "1" (ICSnm: ICPn=1), the interrupt request is generated.
bit3 bit2	EGm1, EGm0: Edge select bits m	For the input capture register m, the trigger edge of the capture operation is set. • Setting of the trigger edge is used to specify enable and stop of the operation. **When set to "00_B":** The operation of input capture is disabled and no edge is detected.
bit1 bit0	EGn1, EGn0: Edge select bits n	For the input capture register n, the trigger edge of the capture operation is set. • Setting of the trigger edge is used to specify enable and stop of the operation. **When set to "00_B":** The operation of input capture is disabled and no edge is detected.

n = 0, 2 m = n + 1

13.3.5 Input Capture Register (IPCP)

Input capture register stores the counter value fetched from 16-bit free-run timer by the capture operation.
The IPCP register is the 16-bit read-only register and has the input capture registers 0 to 3 (IPCP0 to IPCP3).

■ Input Capture Register (IPCP)

Figure 13.3-5 Input Capture Register (IPCP)

Address									Reset value
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
IPCP0 (upper): 007921H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
IPCP0 (lower): 007920H	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
IPCP1 (upper): 007923H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
IPCP1 (lower): 007922H	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
IPCP2 (upper): 007925H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
IPCP2 (lower): 007924H	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
IPCP3 (upper): 007927H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	XXXXXXXXB
	R	R	R	R	R	R	R	R	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
IPCP3 (lower): 007926H	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	XXXXXXXXB
	R	R	R	R	R	R	R	R	

R : Read only
X : Undefined

When the trigger edge of the capture operation (ICSnm: set by EGn1, EGn0 or EGm1, EGm0) is detected by the IN0 to IN3 pins, the counter value of the 16-bit free-run timer is stored in the input capture registers 0 to 3 corresponding to each pin.

However, the input capture registers 0 and 1 can be selected a signal from the LIN-UART as the input signal (ICE: selected by IEI bit). See "13.3.6 Input Capture Edge Register (ICE)" for details.

The input capture register can be read, but can not be written.

n =0,2 m = n+1

Note:

Always use a word instruction (MOVW) to read the input capture register.

13.3.6 Input Capture Edge Register (ICE)

The input capture edge register has a function to indicate the selected edge direction and to select whether the input signal is inputted from either external pin or LIN-UART. By cooperating with the LIN-UART, the baud rate measurement at the LIN slave operation can be performed.

The correspondence between ICE01 to ICE23 / channel name and input pin (UART) name is shown as follows.

ICE01: input capture ch0, ch1	**IN0(/UART0)**	**IN1(/UART1)**
ICE23: input capture ch2, ch3	**IN2**	**IN3**

■ Input Capture Edge Register (ICE)

Figure 13.3-6 Input Capture Edge Register (ICE)

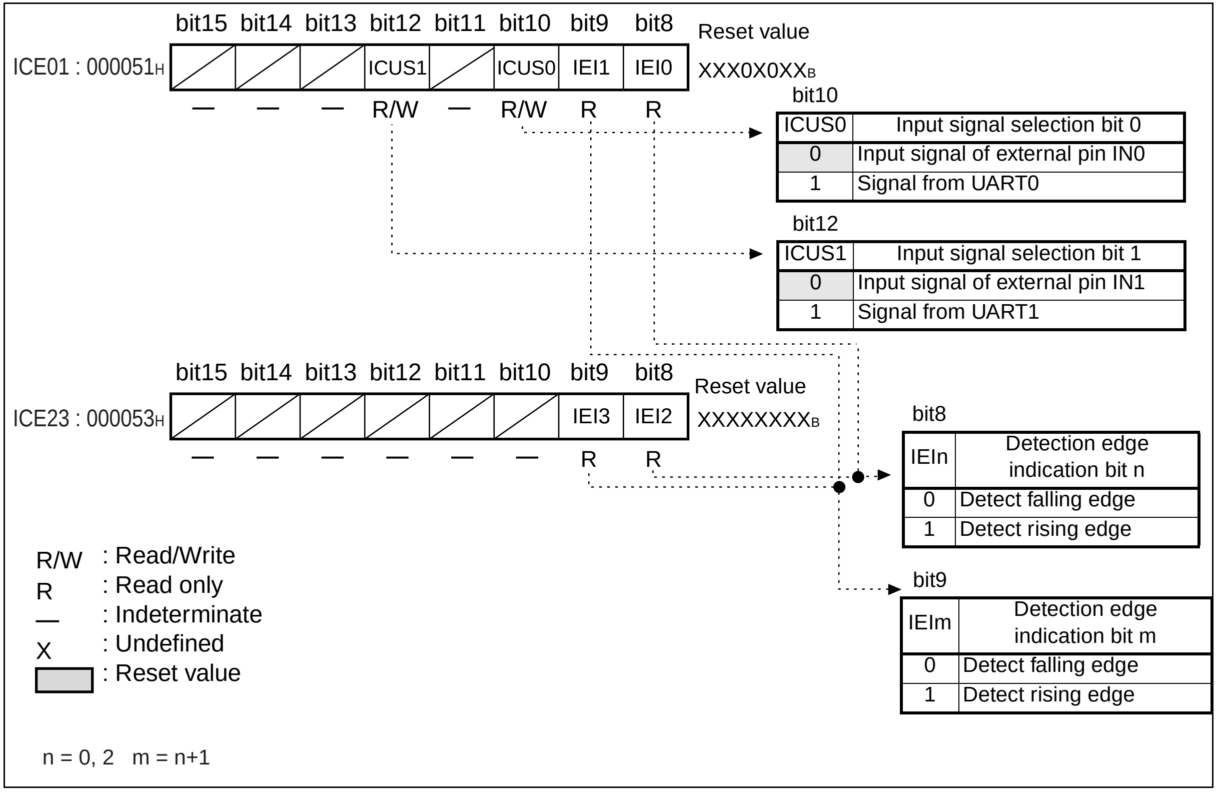


Table 13.3-5 Functions of Input Capture Edge Register 01 (ICE01)

Bit name		Function
bit15 to bit13	Undefined bits	**Read : The value is undefined.** **Write: No effect.**
bit12	ICUS1: Input signal selection bit 1	This bit selects the input signal used as the trigger of the input capture 1. **When set to "0": Select the external pin IN1.** **When set to "1": Select the IN-UART1.**
bit11	Undefined bit	**Read : The value is undefined.** **Write: No effect.**
bit10	ICUS0: Input signal selection bit 0	This bit selects the input signal used as the trigger of the input capture 0. **When set to "0": Select the external pin IN0.** **When set to "1": Select the IN-UART0.**
bit9	IEI1: Detection edge indication bit 1	This bit indicated the edge detected by the input capture 1 (rising/falling). This bit is read only. **"0": Indicate that falling edge is detected.** **"1": Indicate that rising edge is detected.** Note: This bit value is disabled when the capture operation is stopped (ICS01 : EG11, EG10="00").
bit8	IEI0: Detection edge indication bit 0	This bit indicated the edge detected by the input capture 0 (rising/falling). This bit is read only. **"0": Indicate that falling edge is detected.** **"1": Indicate that rising edge is detected.** Note: This bit value is disabled when the capture operation is stopped (ICS01 : EG01, EG00="00").

Table 13.3-6 Functions of Input Capture Edge Register 23 (ICE23)

Bit name		Function
bit15 to bit10	Undefined bits	**Read : The value is undefined.** **Write: No effect.**
bit9	IEI3: Detection edge indication bit 3	This bit indicates the edges detected by the input capture 3 (rising/falling). This bit is read only. **"0": Indicate that falling edge is detected.** **"1": Indicate that rising edge is detected.** Note: This bit value is disabled when the capture operation is stopped (ICSnm : EGm1, EGm0="00"). (n=2, m=n+1)
bit8	IEI2 : Detection edge indication bit 2	This bit indicates the edges detected by the input capture 2 (rising/falling). This bit is read only. **"0": Indicate that falling edge is detected.** **"1": Indicate that rising edge is detected.** Note: This bit value is disabled when the capture operation is stopped (ICS23 : EG21, EG20="00").

Note:

In the input capture 0 and 1, if the input signal is selected to the LIN-UART (ICE01:ICUS), the input capture is used to calculate the baud rate when the LIN-UART operates the LIN slave. In this case, it must be set to the input capture interrupt enable (ICS01:ICE0=1 or ICE1=1) and to the detection of both edges (ICS01:EG01, EG00=11_B or EG11, EG10=11_B). See "20.7.3 Operation with LIN Function (Operation Mode 3)" for details of the baud rate calculation.

13.4 Interrupts of 16-bit I/O Timer

The interrupt factors of the 16-bit I/O timer has overflow of the counter value in the 16-bit free-run timer, trigger edge input to the input capture input pin, and trigger edge input for the LIN slave baud rate measurement from the LIN-UART.

The EI^{2}OS can be started by the interrupt of the input capture.

■ Interrupts of 16-bit I/O Timer

Table 13.4-1 shows interrupt control bits and interrupt factors of 16-bit I/O timer.

Table 13.4-1 Interrupts of 16-bit I/O Timer

	Timer counter overflow interrupt	Input capture interrupt
Interrupt request flag	TCCSL: IVF	ICSnm: ICPn, ICPm
Interrupt request output enable bit	TCCSL: IVFE	ICSnm: ICEn, ICEm
Interrupt factor	Counter overflow of 16-bit free-run timer	Valid edge input to the input capture input pin and trigger edge input for the LIN slave baud rate measurement from the LIN-UART

n = 0, 2 m = n+1

● Timer counter overflow interrupt

When the timer overflow interrupt request is set:

The timer overflow generation flag of the timer control status register is set in the following cases (TCCSL:IVF=1).

- When overflow ("FFFF$_H$" → "0000$_H$") occurs at counting up of the 16-bit free-run timer.

When the timer overflow interrupt request occurs:

When the timer overflow interrupt request is set to enable (TCCSL:IVFE=1) if the timer overflow generation flag is set to "1" (TCCSL:IVF=1), the interrupt request is generated.

● Input capture Interrupt

When the valid edge set by the input capture pin (ICS:EG) is detected, or when the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted (valid edge must be set to both edges), the interrupt is shown below.

- The counter value of the detected 16-bit free-run timer is stored to the input capture register.
- The valid edge detection flag of the input capture control status register is set to "1". (ICS: ICP=1)
- When the output of the input capture interrupt request is set to enable (ICS: ICE=1), the interrupt request is generated.

■ 16-bit I/O Timer Interrupt and EI^{2}OS

Reference:

For details of the interrupt number, interrupt control register, and interrupt vector address, see "CHAPTER 3 INTERRUPTS".

■ Correspondence to EI^{2}OS Function

The input capture corresponds to the EI^{2}OS function.

However, to use the EI^{2}OS function, it is necessary to disable other interrupt that shares the interrupt control register (ICR).

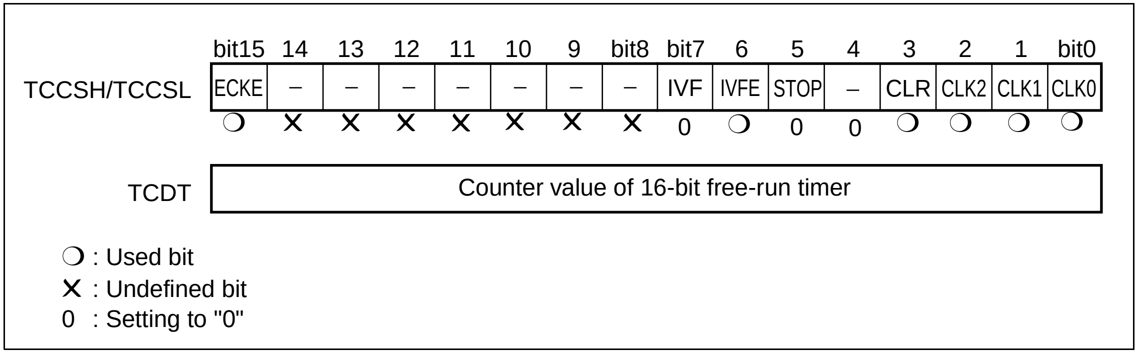
13.5 Explanation of Operation of 16-bit Free-run Timer

After a reset, the 16-bit free-run timer starts incrementing from "0000$_H$". The counter value of the 16-bit free-run timer is the base time of the input capture.

■ Explanation of Operation of 16-bit Free-run Timer

Operation of the 16-bit free-run timer requires the setting shown in Figure 13.5-1 .

Figure 13.5-1 Setting of 16-bit Free-run Timer



	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
TCCSH/TCCSL	ECKE	–	–	–	–	–	–	–	IVF	IVFE	STOP	–	CLR	CLK2	CLK1	CLK0
	❍	✕	✕	✕	✕	✕	✕	✕	0	❍	0	0	❍	❍	❍	❍

TCDT: Counter value of 16-bit free-run timer

❍ : Used bit
✕ : Undefined bit
0 : Setting to "0"

[Setting of counter value in 16-bit free-run timer]

- Because the timer operation is enabled (TCCSL:STOP=0) after a reset, the 16-bit free-run timer starts incrementing from the counter value "0000$_H$".
- When setting the counter value of the 16-bit free-run timer, disable the operation of the 16-bit free-run timer (TCCSL:STOP=1), set the value that starts counting to the timer data register, enable the timer operation (TCCSL:STOP=0).

[Generation of overflow and interrupt request]

- When overflow ("FFFF$_H$" → "0000$_H$") occurs in the 16-bit free-run timer, the timer overflow generation flag is set to "1" (TCCSL:IVF) and starts incrementing from "0000$_H$".
- When the timer overflow interrupt request is enabled (TCCSL:IVFE=1), the interrupt request is occurred.

[Clear factor of counter value and clear timing]

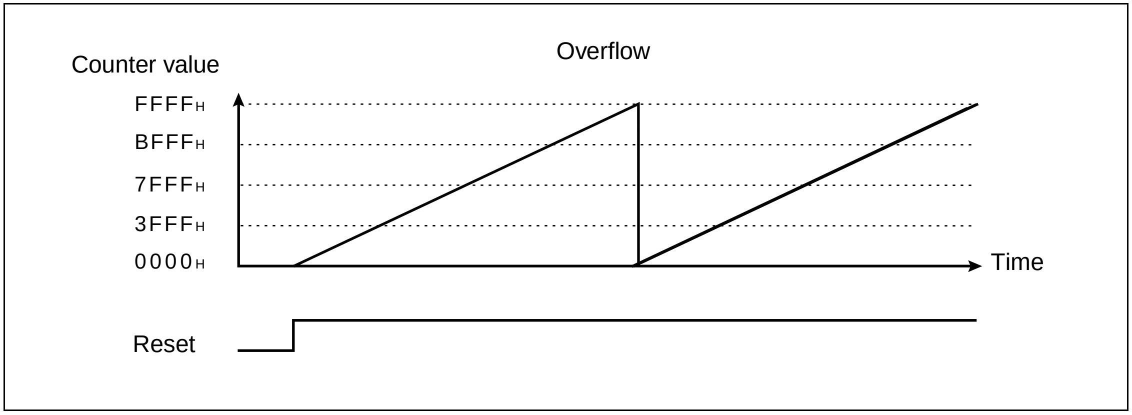
Table 13.5-1 shows the clear factor and clear timing of the 16-bit free-run timer.

Table 13.5-1 Clear Factor of Counter Value and Clear Timing

Clear factor	Clear timing
When "1" to timer clear bit of timer control status register (TCCSL: CLR)	Synchronize with generation of factor
Write "0000$_H$" to timer data register during stopping	Synchronize with generation of factor
Reset	Synchronize with generation of factor
Timer overflow	Synchronize with count timing

Figure 13.5-2 shows counter clearing at an overflow.

Figure 13.5-2 Counter Clearing at an Overflow



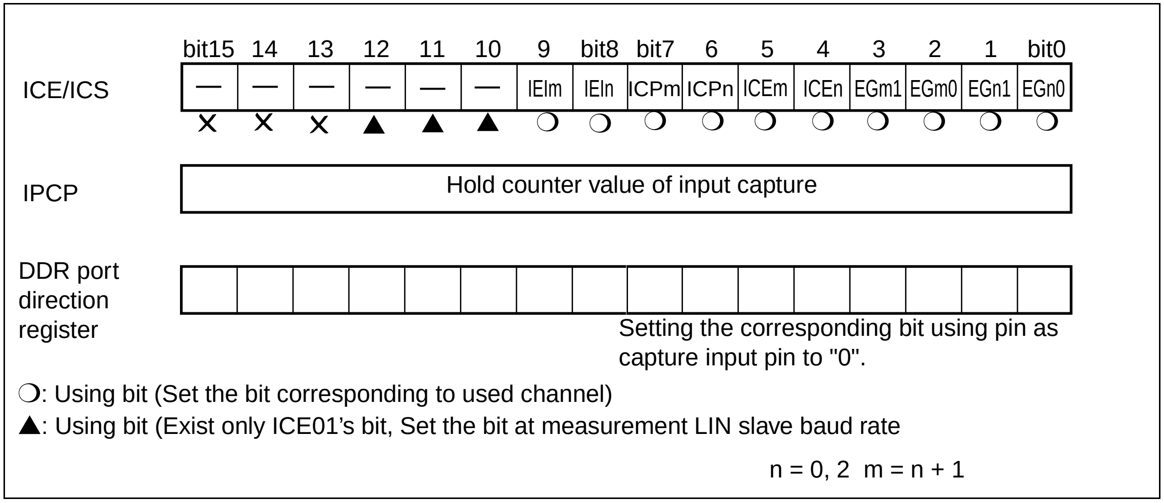
13.6 Explanation of Operation of Input Capture

The input capture stores the counter value of the 16-bit free-run timer to the input capture register at the timing that is detected the input signal of the valid edge from the external input pin or that the trigger edge for the LIN slave baud rate measurement is inputted, the interrupt request is generated.

■ Setting of Input Capture

Operation of the input capture requires the setting shown in Figure 13.6-1

Figure 13.6-1 Setting of Input Capture


bit15 14 13 12 11 10 9 bit8 bit7 6 5 4 3 2 1 bit0
ICE/ICS
— — — — — — IEIm IEIn ICPm ICPn ICEm ICEn EGm1 EGm0 EGn1 EGn0
× × × ▲ ▲ ▲ ❍ ❍ ❍ ❍ ❍ ❍ ❍ ❍ ❍ ❍
IPCP
Hold counter value of input capture
DDR port direction register
Setting the corresponding bit using pin as capture input pin to "0".

❍: Using bit (Set the bit corresponding to used channel)

▲: Using bit (Exist only ICE01's bit, Set the bit at measurement LIN slave baud rate

n = 0, 2 m = n + 1

[Input capture operation]

The following operation is executed when the set valid edge (ICS:EG) is detected in the input capture pin, or when the trigger edge for the LIN slave baud rate measurement from the LIN-UART is inputted.

- The counter value of the 16-bit free-run timer to time detected is stored in the input capture register.
- The detected edge direction is stored to the detection edge indication bit. (rising:IEI=1, falling:IEI=0)
- The valid edge detection flag of the input capture control status register is set to "1". (ICS:ICP=1)
- When the input capture interrupt request is enabled (ICS:ICE=1), the interrupt request is generated.
- To measure the baud rate at the LIN slave operation, it is necessary to set the input signal to the LIN-UART (ICE:ICUS), enable the input capture interrupt request (ICS:ICE=1), and set the valid edge to both edges (ICE:EG1, EG0=11_B). See "20.7.3 Operation with LIN Function (Operation Mode 3)" for the calculation of the baud rate.

Figure 13.6-2 shows the timing of fetching a data for the input capture. Figure 13.6-3 shows the operation when valid edge is set to the rising edge/falling edge. Figure 13.6-4 shows the operation when valid edge is set to both edges.

Figure 13.6-2 Timing of Fetching Data for Input Capture

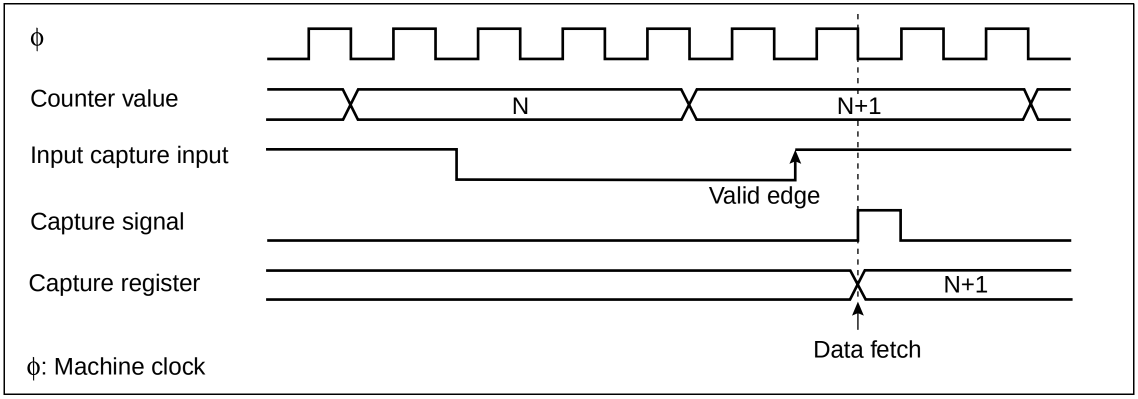


Figure 13.6-3 Operation of Input Capture (Rising edge/falling edge)

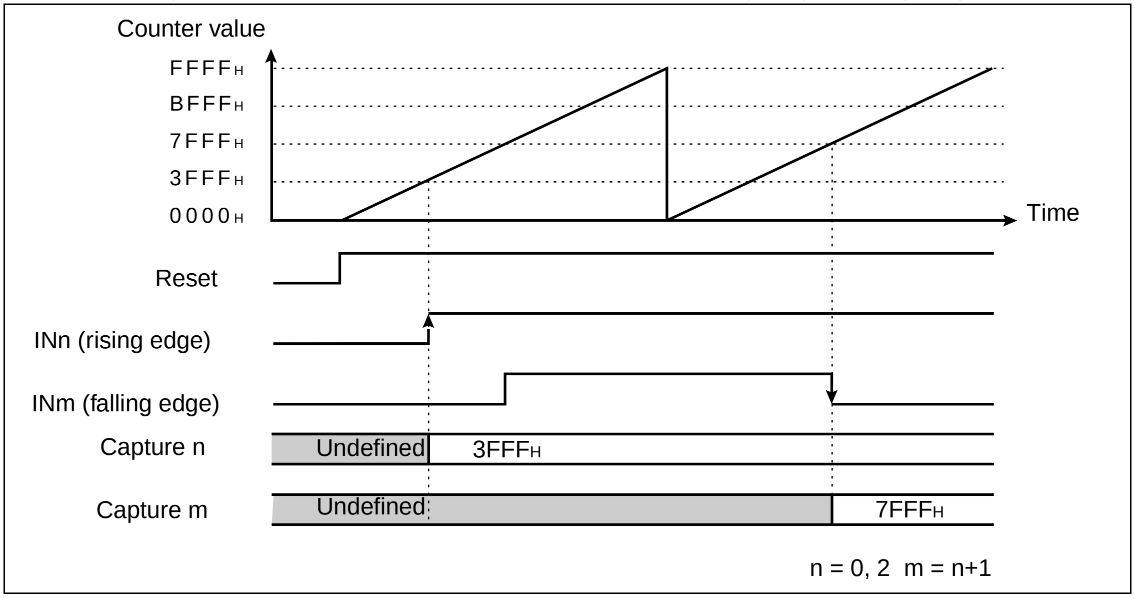
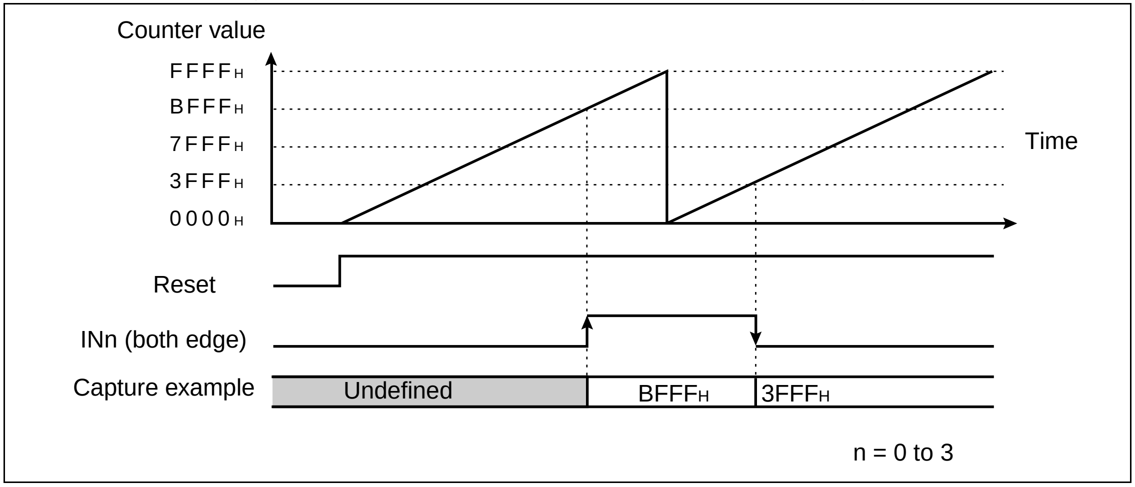


Figure 13.6-4 Operation of Input Capture (both edges)



13.7 Precautions when Using 16-bit I/O Timer

This section explains the precautions when using the 16-bit I/O timer.

■ Precautions when Using 16-bit I/O Timer

● Precautions when setting 16-bit free-run timer

- Do not change the count clock select bits (TCCSL: CLK2, CLK1, CLK0) during the operation in the 16-bit free-run timer (TCCSL: STOP = 0).
- The counter value of the 16-bit free-run timer is cleared to "0000_H" by reset.
- Stop the 16-bit free-run timer (TCCSL:STOP=1), then write the counter value to the timer data register (TCDT) directly.
- Always use a word instruction to write the timer data register (TCDT).

● Operation delay by synchronization operation

The input capture generates delay of operation time because it synchronizes with the operation clock. After the input capture detects the trigger signal from the pin, it synchronizes with the machine clock and performs the capture operation.

13.8 Program Example of 16-bit I/O Timer

This section gives a program example of the 16-bit I/O timer.

■ Program Example of 16-bit I/O Timer

● Processing specification

- The cycle of a signal input to the IN0 pin is measured.
- The 16-bit free-run timer 0 and input capture 0 are used.
- The rising edge is selected as the trigger to be detected.
- The machine clock (ϕ) is 24 MHz and the count clock of the free-run timer is 4/ϕ (0.17 µs).
- The timer overflow interrupt and input capture interrupt of input capture 0 are used.
- The overflow interrupt of the 16-bit free-run timer is counted beforehand and used for the cycle calculation.
- The cycle can be determined from the following equation:

Cycle = (overflow count × "10000_H" + nth IPCP0 value - (n-1)th IPCP0 value) × count clock cycle

= (overflow count × 10000_H + nth IPCP0 value - (n-1)th IPCP0 value) × 0.17 µs

● Coding example

```
ICR09   EQU   0000B9H             ;Interrupt control register
ICR11   EQU   0000BBH             ;Interrupt control register
DDR2    EQU   000012H             ;Port 2 direction register
TCCSL   EQU   007942H             ;Timer control status register
TCDT    EQU   007940H             ;Timer data register
ICS01   EQU   000050H             ;Input capture control status register
IPCP0   EQU   007920H             ;Input capture register 0
IVF0    EQU   TCCSL:7             ;Timer overflow generation flag bit
ICP0    EQU   ICS01:6             ;Valid edge detection flag bit
DATA    DSEG  ABS=00H
        ORG   0100H
OV_CNT  RW    1H
DATA    ENDS                      ;Overflow count counter
;
;---------Main program-----------------------------------------
CODE          CSEG
START:
;                                 ;Stack pointer (SP),
                                  ;already initialized
        AND   CCR,#0BFH           ;Interrupt disable
        MOV   I:ICR09,#00H        ;Interrupt level 0(strongest)
        MOV   I:ICR11,#00H        ;Interrupt level 0(strongest)
        MOV   I:DDR2,#00000000B   ;Port 2 direction setting
        MOV   I:TCCSL,#01001010B  ;Count enable, Counter clear,
                                  ;Overflow, Interrupt enable,
                                  ;Count clock selection, Counter clear
```

```
       MOV   I:ICS01,#00010001B ;IN0 pin selection, External trigger,
                                ;IPCP0 rising edge
                                ;Without IPCP1 edge detection
                                ;Clear each valid edge detection flag
                                ;Input capture interrupt request enable
       MOV   ILM,#07H           ;Set ILM in PS to level 7
       OR    CCR,#40H           ;Interrupt enable
LOOP:
       :
       User processing
       :
       BRA   LOOP
;---------Interrupt program---------------------------------------------
WARI0:
       CLRB  I:ICP0             ;Clear valid edge detection flag
       :                        ;Save OV-CNT and input capture value
       User processing
       :
       MOV   A,0                ;Clear overflow count counter
       MOV   OV_CNT,A           ;for next cycle measurement
       RETI                     ;Recover from interrupt
WARI1:
       CLRB  I:IVF0             ;Clear timer overflow generation flag
       INC   OV_CNT             ;Increment overflow counter
       :
       User processing
       :
       RETI                     ;Recover from interrupt
CODE   ENDS
;---------Vector setting------------------------------------------------
VECT   CSEG  ABS=0FFH
       ORG   00FF78H            ;Setting vector to interrupt number #33($21_H$)
                                ;(Input capture)
       DSL   WARI0
       ORG   00FF84H            ;Setting vector to interrupt number #30($1E_H$)
                                ;(Overflow)
       DSL   WARI1
       ORG   00FFDCH            ;Reset vector setting
       DSL   START
       DB    00H                ;Setting to single-chip mode
VECT   ENDS
       END   START
```

CHAPTER 14

16-BIT RELOAD TIMER

This chapter describes the functions and operation of the 16-bit reload timer.

14.1 Overview of the 16-bit Reload Timer

The 16-bit reload timer has the following functions:

- **The count clock can be selected from three internal clocks and external event clocks.**
- **A software trigger or external trigger can be selected as the start trigger.**
- **If the 16-bit timer register (TMR) underflows, an interrupt can be generated to the CPU. The 16-bit reload timer can be used as an interval timer by using an interrupt.**
- **If the TMR underflows, either the one-shot mode for stopping the TMR count operation, or the reload mode for reloading the value of the 16-bit reload register (TMRLR) to the TMR to continue the TMR count operation can be selected.**
- **The 16-bit reload timer corresponds to the EI^{2}OS (correspond to 2 channels).**
- **The MB90360 series has two channels of 16-bit reload timers.**

■ Operation Modes of 16-bit Reload Timer

Table 14.1-1 indicates the operation modes of the 16-bit reload timer.

Table 14.1-1 Operation Modes of 16-bit Reload Timer

Count clock	Start trigger	Operation performed upon underflow
Internal clock mode	Software trigger External trigger	One-shot mode Reload mode
Event count mode	Software trigger	One-shot mode Reload mode

■ Internal Clock Mode

- When the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) are set to "00_B", "01_B" or "10_B", the 16-bit reload timer is set in the internal clock mode.
- In the internal clock mode, the 16-bit reload timer decrements in synchronization with the internal clock.
- The count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) can be used to select three count clock cycles.
- The start trigger sets the edge detection for a software trigger or an external trigger.

■ Event Count Mode

- When the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) are set to "11_B", the 16-bit reload timer is set to the event count mode.
- In the event count mode, the 16-bit reload timer decrements in synchronization with the edge detection of the external event clock input to the TIN pin.
- A software trigger is selected as the start trigger.
- The 16-bit reload timer can be used as an interval timer by using a fixed cycle of the external clock.

■ Operation at Underflow

When the start trigger is inputted, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register, starting decrementing in synchronization with the count clock. When the 16-bit timer register (TMR) is decremented from "0000_H" to "$FFFF_H$", an underflow occurs.

- When an underflow occurs with an underflow interrupt enabled (TMCSR:INTE = 1), an underflow interrupt is generated.
- The 16-bit reload timer operation when an underflow occurs is set by the reload select bit in the timer control status register (TMCSR:RELD).

[One-shot mode (TMCSR: RELD=0)]

When an underflow occurs, the TMR count operation is stopped. When the next start trigger is inputted, the value set in the TMRLR is reloaded in the TMR, starting the TMR count operation.

- In the one-shot mode, during the TMR count operation, a High-level or Low-level rectangular wave is outputted from the TOT pin.
- The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of the rectangular wave.

[Reload mode (TMCSR: RELD=1)]

When an underflow occurs, the value set in the TMRLR is reloaded to the TMR, continuing the TMR count operation.

- In the reload mode, a toggle wave inverting the output level of the TOT pin is outputted each time an underflow occurs during the TMR count operation.
- The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of a toggle wave at starting the reload timer.
- The 16-bit reload timer can be used as an interval timer by using an underflow interrupt.

Table 14.1-2 Interval Time for the 16-bit Reload Timer

Count clock	Count clock period	Interval time
Internal clock mode	2^1T(0.083 µs)	0.083 µs to 5.46 ms
	2^3T(0.33 µs)	0.33 µs to 21.8 ms
	2^5T(1.3 µs)	1.3 µs to 87.4 ms
Event count mode	2^3T or more	0.33 µs or more

T: Machine cycle
The values in interval time and the parenthesized values are provided when the machine clock operates at 24 MHz.

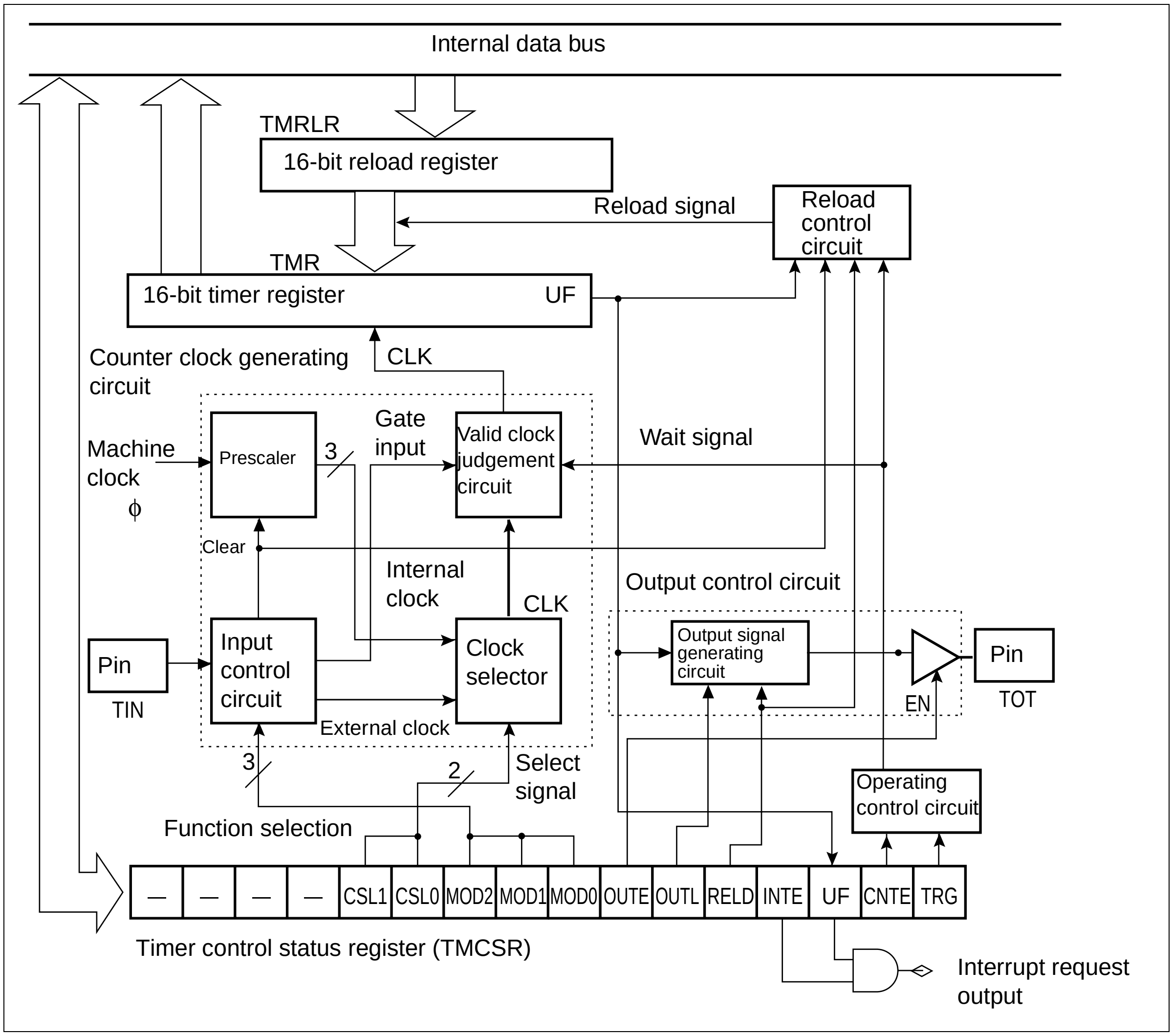
14.2 Block Diagram of 16-bit Reload Timer

The 16-bit reload timers 2 and 3 composed of the following seven blocks:

- **Count clock generator**
- **Reload controller**
- **Output controller**
- **Operation controller**
- **16-bit timer register (TMR)**
- **16-bit reload register (TMRLR)**
- **Timer control status register (TMCSR)**

■ Block Diagram of 16-bit Reload Timer

Figure 14.2-1 Block Diagram of 16-bit Reload Timer



- Details of pins in block diagram

There are two channels for 16-bit reload timer.

The actual pin names, outputs to resources, and interrupt request numbers for each channel are as follows:

Table 14.2-1 Pin Names, Outputs to Resources, and Interrupt Request Numbers of 16-bit Reload Timer

	Reload timer 2	Reload timer 3
TIN pin	P82	P53
TOT pin	P83	P54
Output to resources	-	-
Interrupt request number	#19(13$_H$)	#20(14$_H$)

- Count clock generator

The count clock generator generates a count clock supplied to the 16-bit timer register (TMR) on the basis of the machine clock or external event clock.

- Reload controller

When the 16-bit reload timer starts operation or the TMR underflows, the reload controller reloads the value set in the 16-bit reload register (TMRLR) to the TMR.

- Output controller

The output controller inverts and enables or disables the output of the TOT pin at underflow.

- Operation controller

The operation controller starts or stops the 16-bit reload timer.

- 16-bit timer register (TMR)

The 16-bit timer register (TMR) is a 16-bit down counter. At read, the value being counted is read.

- 16-bit reload register (TMRLR)

The 16-bit reload register (TMRLR) sets the interval time of the 16-bit reload timer. When the 16-bit reload timer starts operation or the 16-bit timer register (TMR) underflows, the value set in the TMRLR is reloaded to the TMR.

- Timer control status register (TMCSR)

The timer control status register (TMCSR) selects the operation mode, sets the operation conditions, selects the start trigger, performs a start using the software trigger, selects the reload operation mode, enables or disables an interrupt request, sets the output level of the TOT pin, and sets the TOT output pin of the 16-bit reload timer.

14.3 Configuration of 16-bit Reload Timer

This section explains the pins, registers, and interrupt factors of the 16-bit reload timer.

■ Pins of 16-bit Reload Timer

The pins of the 16-bit reload timer serve as general-purpose I/O ports. Table 14.3-1 shows the pin functions and the pin settings required to use the 16-bit reload timer.

Table 14.3-1 Pins of 16-bit Reload Timer

Pin name	Pin function	Pin Setting Required for Use in 16-bit Reload Timer
P82 / SIN0 / INT14R / TIN2	General-purpose I/O port/ UART input 0/ External interrupt 14/ 16-bit reload timer input 2	• Port direction register: Setting for the input port (DDR8:D82=0) • Serial control register: Setting for the reception disable (SCR0:RXE=0) • Disable the external interrupt (external interrupt enable register ENIR1: EN14 = 0)
P83 / SOT0 / TOT2	General-purpose I/O port/ UART output 0/ 16-bit reload timer output 2	• Serial control register : Setting for the transmission disable (SCR0:TXE=0) • Timer control status register: Enable the timer output (TMCSR2: OUTE=1)
P53 / AN11 / TIN3	General-purpose I/O port/ A/D converter analog input 11/ 16-bit reload timer input 3	• Port direction register : Setting for the input port (DDR5:D53=0) • Analog input enable register: Setting for the prohibition (ADER5:ADE11=0)
P54 / AN12 / TOT3	General-purpose I/O port/ A/D converter analog input 12/ 16-bit reload timer output 3	• Analog input enable register: Setting for the prohibition (ADER5:ADE12=0) • Timer control status register: Enable the timer output (TMCSR3: OUTE=1)

■ 16-bit Reload Timer Registers and Reset Value

- ● 16-bit reload timer 2 register

Figure 14.3-1 List of 16-bit Reload Timer 2 Register and Reset Value

	bit	15	14	13	12	11	10	9	8
Timer Control Status Register Upper (TMCSR2)		X	X	X	X	0	0	0	0
	bit	7	6	5	4	3	2	1	0
Timer Control Status Register Lower (TMCSR2)		0	0	0	0	0	0	0	0
	bit	15	14	13	12	11	10	9	8
16-bit Timer Register Upper (TMR2)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Timer Register Lower (TMR2)		X	X	X	X	X	X	X	X
	bit	15	14	13	12	11	10	9	8
16-bit Reload Register Upper (TMRLR2)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Reload Register Lower (TMRLR2)		X	X	X	X	X	X	X	X

X : Undefined

- ● 16-bit reload timer 3 register

Figure 14.3-2 List of 16-bit Reload Timer 3 Register and Reset Value

	bit	15	14	13	12	11	10	9	8
Timer Control Status Register Upper (TMCSR3)		X	X	X	X	0	0	0	0
	bit	7	6	5	4	3	2	1	0
Timer Control Status Register Lower (TMCSR3)		0	0	0	0	0	0	0	0
	bit	15	14	13	12	11	10	9	8
16-bit Timer Register Upper (TMR3)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Timer Register Lower (TMR3)		X	X	X	X	X	X	X	X
	bit	15	14	13	12	11	10	9	8
16-bit Reload Register Upper (TMRLR3)		X	X	X	X	X	X	X	X
	bit	7	6	5	4	3	2	1	0
16-bit Reload Register Lower (TMRLR3)		X	X	X	X	X	X	X	X

X : Undefined

■ Generation of Interrupt Request from 16-bit Reload Timer

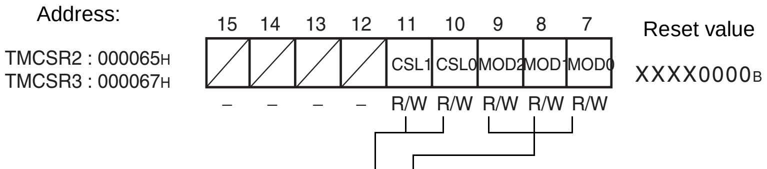
When the 16-bit reload timer is started and the count value of the 16-bit timer register is decremented from "0000_H" to "$FFFF_H$", an underflow occurs. When an underflow occurs, the UF bit in the timer control status register is set to 1 (TMCSR:UF). If an underflow interrupt is enabled (TMCSR:INTE = 1), an interrupt request is generated.

14.3.1 Timer Control Status Registers (High) (TMCSR:H)

The timer control status registers (High) (TMCSR:H) set the operation mode and count clock.
This section also explains the bit 7 in the timer control status registers (Low) (TMCSR:L).

■ Timer Control Status Registers (High) (TMCSR:H)

Figure 14.3-3 Timer Control Status Registers (High) (TMCSR:H)



bit9	bit8	bit7	Operating mode select bit (internal clock mode) (CSL1, 0="00$_B$", "01$_B$", "10$_B$")	
MOD2	MOD1	MOD0	Input pin function	Valid edge, level
0	0	0	Trigger disable	–
0	0	1	Trigger input	Rising edge
0	1	0		Falling edge
0	1	1		Both edges
1	X	0	Gate input	"L" level
1	X	1		"H" level

bit9	bit8	bit7	Operating mode select bit (event count mode) (CSL1, 0="11$_B$")	
MOD2	MOD1	MOD0	Input pin function	Valid edge
X	0	0	–	–
X	0	1	Trigger input	Rising edge
X	1	0		Falling edge
X	1	1		Both edges

bit11	bit10	Count clock select bit	
CSL1	CSL0	Count clock	Count clock cycle
0	0	Internal clock mode	2^1T
0	1		2^3T
1	0		2^5T
1	1	Event count mode	External event clock

T: Machine cycle

R/W	: Read/Write
X	; Indeterminate
–	: Undefined
▨	: Reset value

Table 14.3-2 Functions of Timer Control Status Registers (High) (TMCSR: H)

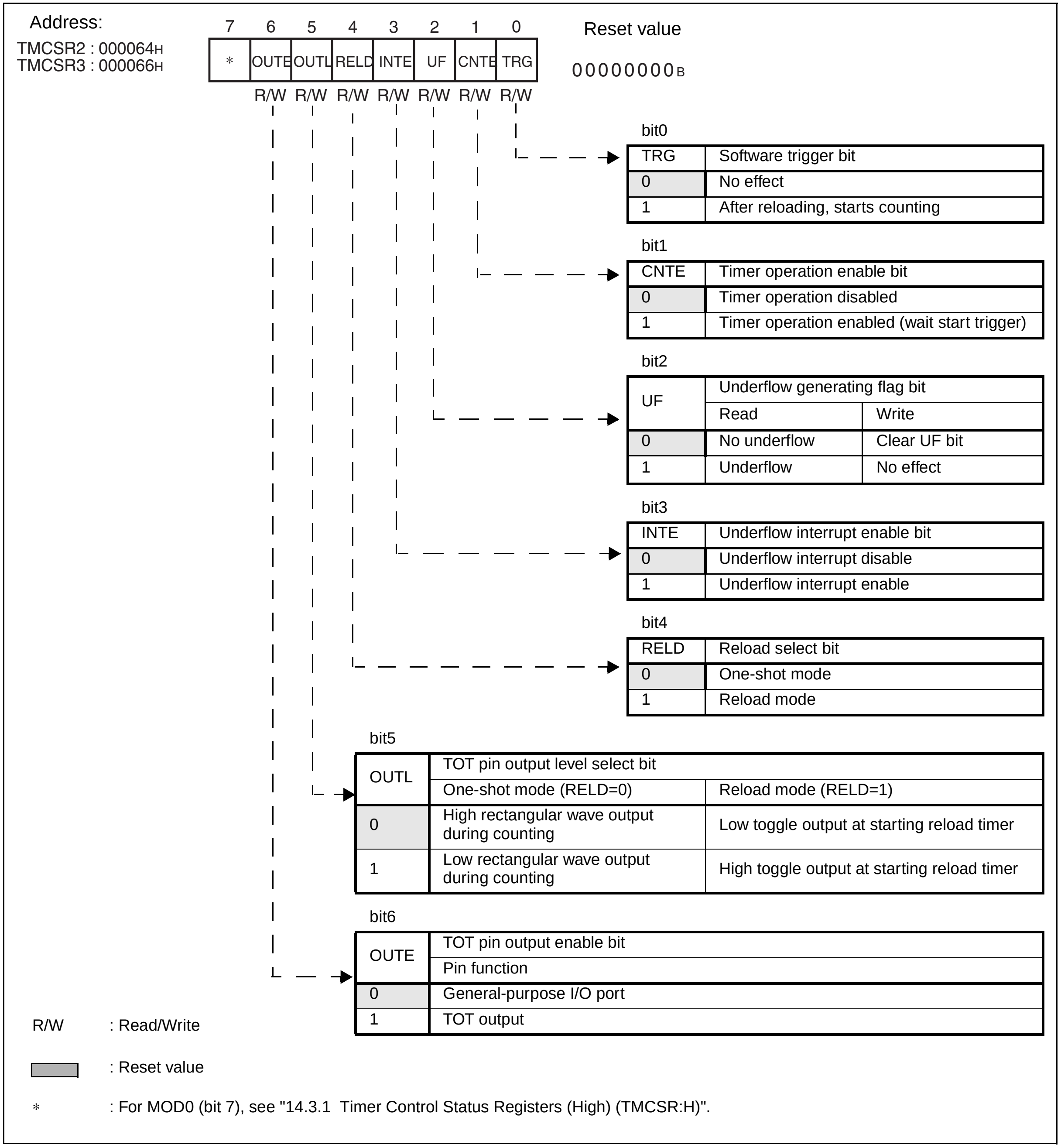
Bit name		Function
bit15 to bit12	Undefined bits	Read: The value is undefined. Write: No effect
bit11 bit10	CSL1, CSL0: Count clock select bits	These bits select the count clock of the 16-bit reload timer. **When set to anything other than "11_B":** These bit is counted by internal clock (internal clock mode). **When set to "11_B":** The edge of the external event clock is counted (event count mode).
bit9 to bit7	MOD2, MOD1, MOD0: Operating mode select bits	These bits set the operation conditions of the 16-bit reload timer. **[Internal clock mode]** The MOD2 bit is used to select the function of the input pin. **When MOD2 bit set to 0:** The input pin functions as a trigger input. The MOD1 and MOD0 bits are used to select the edge to be detected. When the edge is detected, the value set in the 16-bit reload register (TMRLR) is reloaded in the 16-bit timer register (TMR), starting the count operation of the TMR. **When MOD2 set to 1:** The input pin functions as a gate input. The MOD1 bit is not used. The MOD0 bit is used to select the signal level (High or Low) to be detected. The count operation of the 16-bit timer register (TMR) is performed only when the signal level is inputted. **[Event count mode]** The MOD2 bit is not used. An external event clock is inputted from the input pin. The MOD1 and MOD0 bits are used to select the edge to be detected.

14.3.2 Timer Control Status Registers (Low) (TMCSR: L)

The timer control status registers (Low) (TMCSR:L) enables or disable the timer operation, check the generation of a software trigger or an underflow, enables or disable an underflow interrupt, select the reload mode, and set the output of the TOT pin.

■ Timer Control Status Registers (Low) (TMCSR: L)

Figure 14.3-4 Timer Control Status Registers (Low) (TMCSR: L)



Address:
TMCSR2 : 000064H
TMCSR3 : 000066H

	7	6	5	4	3	2	1	0	Reset value
	*	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG	00000000B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	

bit0

TRG	Software trigger bit
0	No effect
1	After reloading, starts counting

bit1

CNTE	Timer operation enable bit
0	Timer operation disabled
1	Timer operation enabled (wait start trigger)

bit2

UF	Underflow generating flag bit	
	Read	Write
0	No underflow	Clear UF bit
1	Underflow	No effect

bit3

INTE	Underflow interrupt enable bit
0	Underflow interrupt disable
1	Underflow interrupt enable

bit4

RELD	Reload select bit
0	One-shot mode
1	Reload mode

bit5

OUTL	TOT pin output level select bit	
	One-shot mode (RELD=0)	Reload mode (RELD=1)
0	High rectangular wave output during counting	Low toggle output at starting reload timer
1	Low rectangular wave output during counting	High toggle output at starting reload timer

bit6

OUTE	TOT pin output enable bit
	Pin function
0	General-purpose I/O port
1	TOT output

R/W : Read/Write

▨ : Reset value

* : For MOD0 (bit 7), see "14.3.1 Timer Control Status Registers (High) (TMCSR:H)".

Table 14.3-3 Timer Control Status Registers (Low) (TMCSR: L)

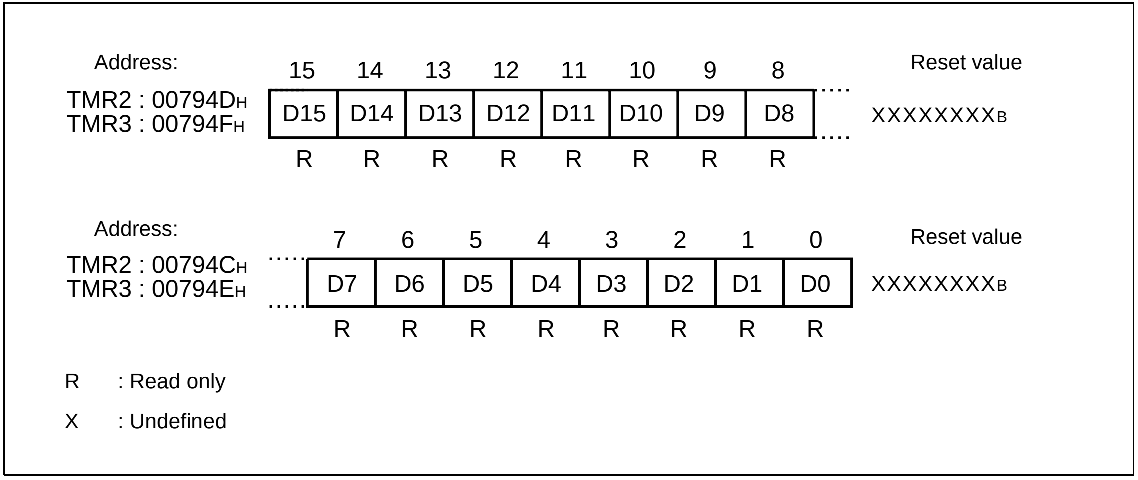
Bit Name		Function
bit6	OUTE: TOT pin Output enable bit	This bit sets the function of the TOT pin of the 16-bit reload timer. **When set to 0:** Functions as general-purpose I/O port **When set to 1:** Functions as TOT pin of 16-bit reload timer
bit5	OUTL: TOT Pin output level select bit	This bit sets the output level of the output pin of the 16-bit reload timer. **<One-shot mode (RELD = 0)>** **When set to 0:** Outputs High-level rectangular wave during TMR count operation **When set to 1:** Outputs Low-level rectangular wave during TMR count operation **<Reload mode (RELD = 1)>** **When set to 0:** Outputs Low-level toggle wave when 16-bit reload timer started **When set to 1:** Outputs High-level toggle wave when 16-bit reload timer started
bit4	RELD: Reload select bit	This bit sets the reload operation at underflow. **When set to 1:** At underflow, reloads value set in TMRLR to TMR, continuing count operation (reload mode) **When set to 0:** At underflow, stops count operation (one-shot mode)
bit3	INTE: Underflow interrupt enable bit	This bit enables or disables an underflow interrupt. When an underflow occurs (TMCSR:UF = 1) with an underflow interrupt enabled (TMCSR:INTE = 1), an interrupt request is generated.
bit2	UF: Underflow generating flag bit	This bit indicates that the TMR underflows. **When set to 0:** Clears this bit **When set to 1:** No effect **Read by read modify write instructions:** 1 is always read.
bit1	CNTE: Timer operation enable bit	This bit enables or disables the operation of the 16-bit reload timer. **When set to 1:** 16-bit reload timer enters start trigger wait state. When the start trigger is inputted, the count operation of the TMR is restarted. **When set to 0:** Stops count operation
bit0	TRG: Software trigger bit	This bit starts the 16-bit reload timer by software. The software trigger function works only when the timer operation is enabled (CNTE = 1). **When set to 0:** Disabled. The state remains unchanged. **When set to 1:** Reloads value set in 16-bit reload register (TMRLR) to 16-bit timer register (TMR), starting TMR count operation **Read:** 0 is always read.

14.3.3 16-bit Timer Registers (TMR)

The 16-bit timer registers are 16-bit down counters. At read, the value being counted is read.

■ 16-bit Timer Registers (TMR)

Figure 14.3-5 16-bit Timer Registers (TMR)



When the timer operation is enabled (TMCSR:CNTE = 1) and the start trigger is inputted, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR), starting the TMR count operation.

When the timer operation is disabled (TMCSR:CNTE = 0), the TMR value is retained.

When the TMR value is counted down from "0000_H" to "$FFFF_H$" during the TMR count operation, an underflow occurs.

[Reload mode]

When the TMR underflows, the value set in the TMRLR is reloaded to the TMR, restarting the TMR count operation.

[One-shot mode]

When the TMR underflows, the TMR count operation is stopped, entering the start trigger input wait state. The TMR value is retained to "$FFFF_H$".

Notes:

- The TMR can be read during the TMR count operation. However, always use the word instruction (MOVW).
- The TMR and the TMRLR are assigned to the same address. At write, the set value can be written to the TMRLR without affecting the TMR. At read, the TMR value being counted can be read.

14.3.4 16-bit Reload Registers (TMRLR)

The 16-bit reload registers set the value to be reloaded to the 16-bit timer register (TMR). When the start trigger is inputted, the value set in the 16-bit reload registers is reloaded to the TMR, starting the TMR count operation.

■ 16-bit Reload Registers (TMRLR)

Figure 14.3-6 16-bit Reload Registers (TMRLR)

Address:
TMRLR2 : 00794DH
TMRLR3 : 00794FH

15	14	13	12	11	10	9	8	Reset value
D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
W	W	W	W	W	W	W	W	

Address:
TMRLR2 : 00794CH
TMRLR3 : 00794EH

7	6	5	4	3	2	1	0	Reset value
D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
W	W	W	W	W	W	W	W	

W : Write only
X : Undefined

Set the 16-bit reload registers after disabling the timer operation (TMCSR:CNTE = 0). After completing setting of the 16-bit reload registers, enable the timer operation (TMCSR:CNTE = 1).

When the start trigger is inputted, the value set in the TMRLR is reloaded to the TMR, starting the TMR count operation.

Notes:

- Perform a write to the TMRLR after disabling the operation of the 16-bit reload timer (TMCSR:CNTE = 0). Always use the word instruction (MOVW).
- The TMRLR and the TMR are assigned to the same address. At write, the set value can be written to the TMRLR without affecting the TMR. At read, the TMR value being counted is read.
- Instructions, such as the INC/DEC instruction, which provide the read modify write (RMW) operation cannot be used.

14.4 Interrupts of 16-bit Reload Timer

The 16-bit reload timer generates an interrupt request when the 16-bit timer register (TMR) underflows.

■ Interrupts of 16-bit Reload Timer

When the value of the TMR is decremented from "0000_H" to "$FFFF_H$" during the TMR count operation, an underflow occurs. When an underflow occurs, the underflow generating flag bit in the timer control status register (TMCSR:UF) is set to l. When an underflow interrupt is enabled (TMCSR:INTE = 1), an interrupt request is generated.

Table 14.4-1 Interrupt Control Bits and Interrupt Factors of 16-bit Reload Timer

	16-bit Reload Timer 2	16-bit Reload Timer 3
Interrupt request flag bit	TMCSR2: UF	TMCSR3: UF
Interrupt request enable bit	TMCSR2: INTE	TMCSR3: INTE
Interrupt factor	Underflow in TMR2	Underflow in TMR3

■ Correspondence between 16-bit Reload Timer Interrupt and EI^2OS

Reference: For details of the interrupt number, interrupt control register, and interrupt vector address, see "CHAPTER 3 INTERRUPTS".

■ EI^2OS Function of 16-bit Reload Timer

The 16-bit reload timers 2 and 3 correspond to the EI^2OS function. An underflow in the TMR starts the EI^2OS.

However, the EI^2OS is available only when other resources sharing the interrupt control register (ICR) do not use interrupts. The 16-bit reload timers 2 and 3 share the ICR04. When using the EI^2OS in the 16-bit reload timers 2 and 3, it is necessary to disable the interrupt of the 16-bit reload timer sharing the interrupt control register.

14.5 Explanation of Operation of 16-bit Reload Timer

This section explains the setting of the 16-bit reload timer and the operation state of the counter.

■ Setting of 16-bit Reload Timer

- Setting of internal clock mode

Counting the internal clock requires the setting shown in Figure 14.5-1 .

Figure 14.5-1 Setting of Internal Clock Mode

	bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	bit0
TMCSR	–	–	–	–	CSL1	CSL0	MOD2	MOD1	MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
					Other than "11_B"		◎	◎	◎	◎	◎	◎	◎	◎	1	◎
TMRLR	Set the reload value to 16-bit timer register															

◎ : Used bit
1 : Set "1".

- Setting of event count mode

Inputting an external event to operate the 16-bit reload timer requires the setting shown in Figure 14.5-2 .

Figure 14.5-2 Setting of Event Count Mode

	bit15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	bit0
TMCSR	–	–	–	–	CSL1	CSL0	MOD2	MOD1	MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
					1	1	❍	❍	❍	❍	❍	❍	❍	❍	1	❍
TMRLR	Set the reload value to 16-bit timer register															

Set the bit of DDR (port direction register) corresponding to the pin to be used as TIN pin to "0".

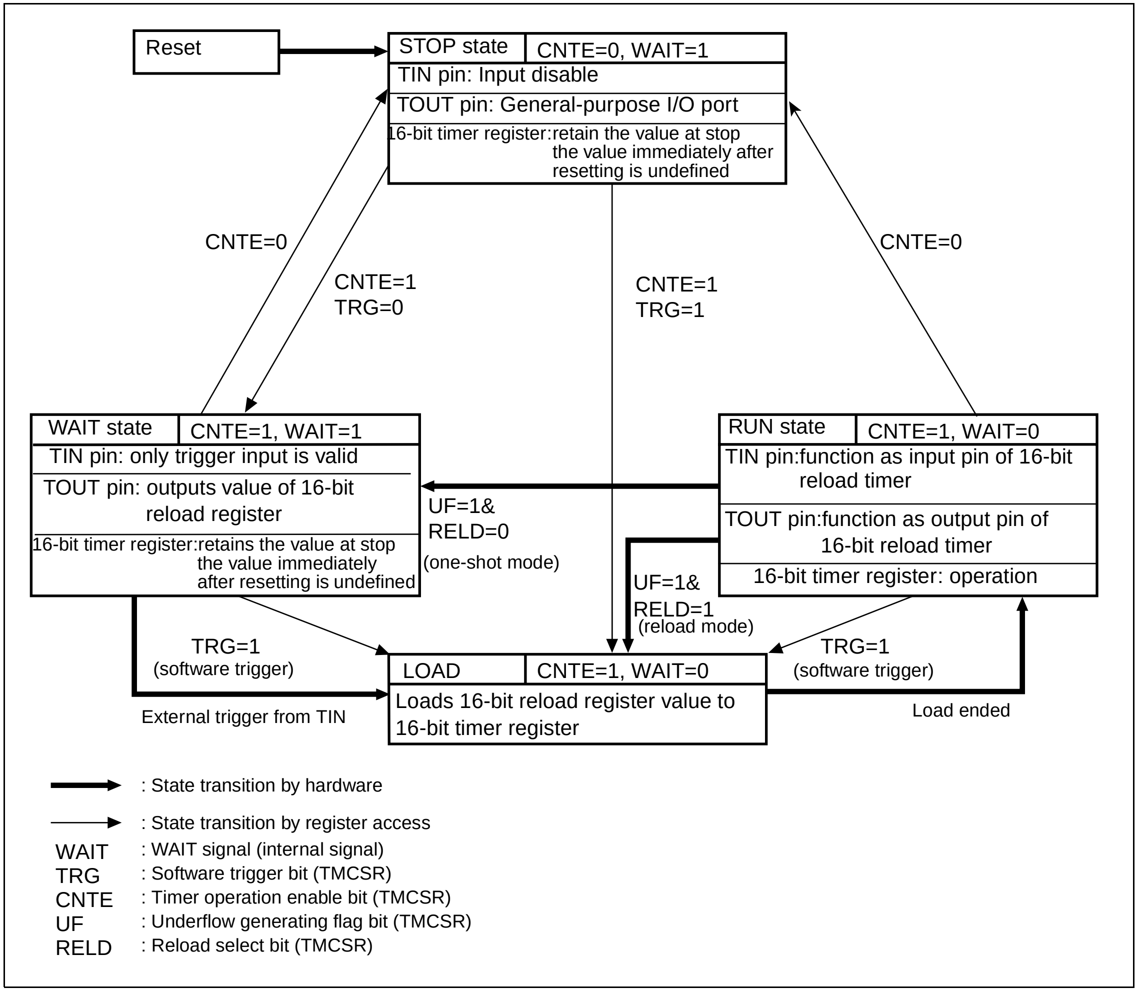
❍: Used bit
1 : Set "1".

■ Operating State of 16-bit Timer Register

The operating state of the 16-bit timer register is determined by the timer operation enable bit in the timer control status register (TMCSR:CNTE) and the WAIT signal. The operating states include the stop state, start trigger input wait state (WAIT state), and RUN state.

Figure 14.5-3 shows the state transition diagram for the 16-bit timer registers.

Figure 14.5-3 State Transition Diagram


Reset
STOP state
CNTE=0, WAIT=1
TIN pin: Input disable
TOUT pin: General-purpose I/O port
16-bit timer register:retain the value at stop
the value immediately after resetting is undefined
CNTE=0
CNTE=1
TRG=0
CNTE=1
TRG=1
CNTE=0
WAIT state
CNTE=1, WAIT=1
TIN pin: only trigger input is valid
TOUT pin: outputs value of 16-bit reload register
16-bit timer register:retains the value at stop
the value immediately after resetting is undefined
UF=1&
RELD=0
(one-shot mode)
RUN state
CNTE=1, WAIT=0
TIN pin:function as input pin of 16-bit reload timer
TOUT pin:function as output pin of 16-bit reload timer
16-bit timer register: operation
UF=1&
RELD=1
(reload mode)
TRG=1
(software trigger)
TRG=1
(software trigger)
LOAD
CNTE=1, WAIT=0
Loads 16-bit reload register value to 16-bit timer register
External trigger from TIN
Load ended
: State transition by hardware
: State transition by register access
WAIT : WAIT signal (internal signal)
TRG : Software trigger bit (TMCSR)
CNTE : Timer operation enable bit (TMCSR)
UF : Underflow generating flag bit (TMCSR)
RELD : Reload select bit (TMCSR)

14.5.1 Operation in Internal Clock Mode

In the internal clock mode, three operation modes can be selected by setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0).When the operation mode and reload mode are set, a rectangular wave or a toggle wave is outputted from the TOT pin.

■ Setting of Internal Clock Mode

- By setting the count clock select bits (CSL1, CSL0) in the timer control status register to "00_B", "01_B" and "10_B", the 16-bit reload timer (TMRLR) is set to the internal clock mode.
- In the internal clock mode, the 16-bit timer register (TMR) decrements in synchronization with the internal clock.
- In the internal clock mode, three count clock cycles can be selected by setting the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0).

[Setting a reload value to TMR]

After the 16-bit reload timer is started, the value set in the TMRLR is reloaded to the TMR.

1. Disables the timer operation (TMCSR:CNTE = 0).
2. Sets a reload value to the TMR in the TMRLR.
3. Enables the timer operation (TMCSR:CNTE = 1).

Note:

It takes 1 T (T: machine cycle (time) to load the value set in the TMRLR to the TMR after the start trigger is inputted.

■ Operation as 16-bit Timer Register Underflows

When the value of the 16-bit timer register (TMR) is decremented from "0000_H" to "$FFFF_H$" during the TMR count operation, an underflow occurs.

- When an underflow occurs, the underflow generating flag bit in the timer control status register (TMCSR:UF) is set to 1.
- When the underflow interrupt enable bit in the timer control status register (TMCSR:INTE) is set to 1, an underflow interrupt is generated.
- The reload operation when an underflow occurs is set by the reload select bit in the timer control status register (TMCSR:RELD).

[One-shot mode (TMCSR:RELD = 0)]

When an underflow occurs, the count operation of the TMR is stopped, entering the start trigger input wait state. When the next start trigger is inputted, the TMR count operation is restarted.

In the one-shot mode, a rectangular wave is outputted from the TOT pin during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of a rectangular wave.

[Reload mode (TMCSR:RELD = 1)]

When an underflow occurs, the value set in the 16-bit reload timer register (TMRLR) is reloaded to the TMR, continuing the TMR count operation.

In the reload mode, a toggle wave inverting the output level of the TOT pin is outputted each time an underflow occurs during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of a toggle wave as the 16-bit reload timer is started.

■ Operation in Internal Clock Mode

In the internal clock mode, the operation mode select bits in the timer control status register (TMCSR:MOD2 to MOD0) can be used to select the operation mode. Disable the timer operation by setting the timer operation enable bit in the timer control status register (TMCSR:CNTE to 0).

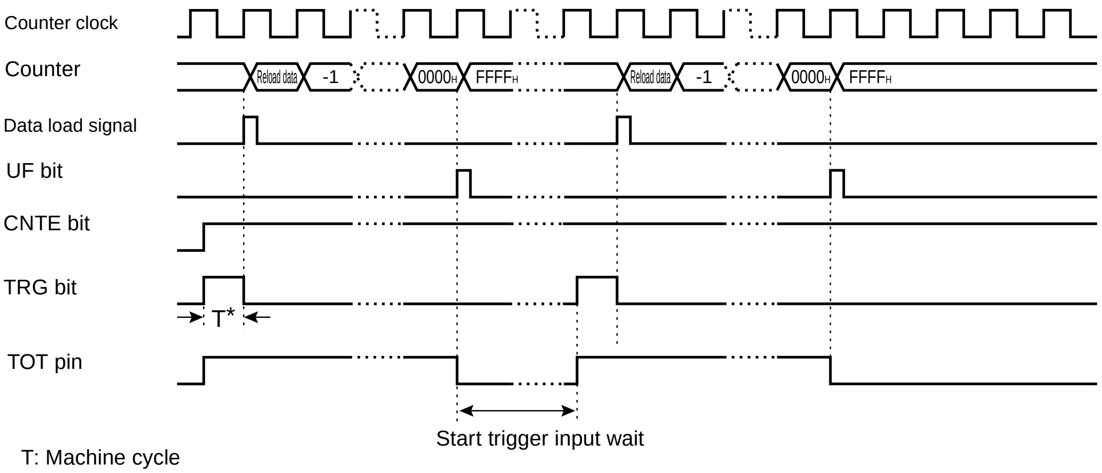
[Software trigger mode (MOD2 to MOD0="000_B")]

If the software trigger mode is set, start the 16-bit reload timer by setting the software trigger bit in the timer control status register (TMCSR:TRG) to 1. When the 16-bit reload timer is started, the value set in the TMRLR is reloaded to the TMR, starting the TMR count operation.

Note:

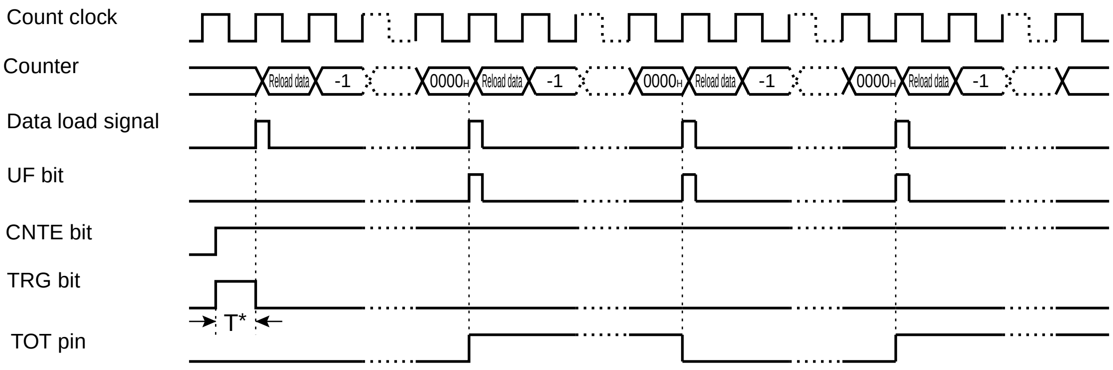
When both the timer operation enable bit in the timer control status register (TMCSR:CNTE) and the software trigger bit in the timer control status register (TMCSR:TRG) are set to 1, the 16-bit reload timer and the count operation of the TMR are started simultaneously.

Figure 14.5-4 Count Operation in Software Trigger Mode (One-shot Mode)



T: Machine cycle
* : It takes 1 machine cycle (time) to load data of reload register from trigger input.

Figure 14.5-5 Count Operation in Software Trigger Mode (Reload Mode)



T: Machine cycle
* : It takes 1 T time to load data of reload register from trigger input.

[External trigger mode (MOD2 to MOD0="001_B", "010_B", "011_B")]

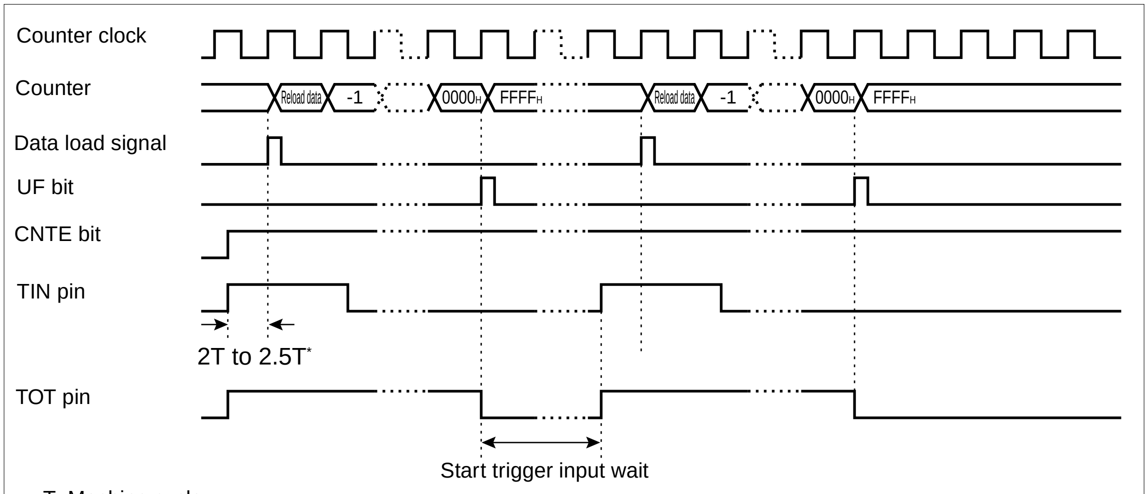
When the external trigger mode is set, the 16-bit reload timer is started by inputting the external valid edge to the TIN pin. When the 16-bit reload timer is started, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR), starting the TMR count operation.

By setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0), the detected edge can be selected from the rising edge, falling edge, and both edges.

Note:

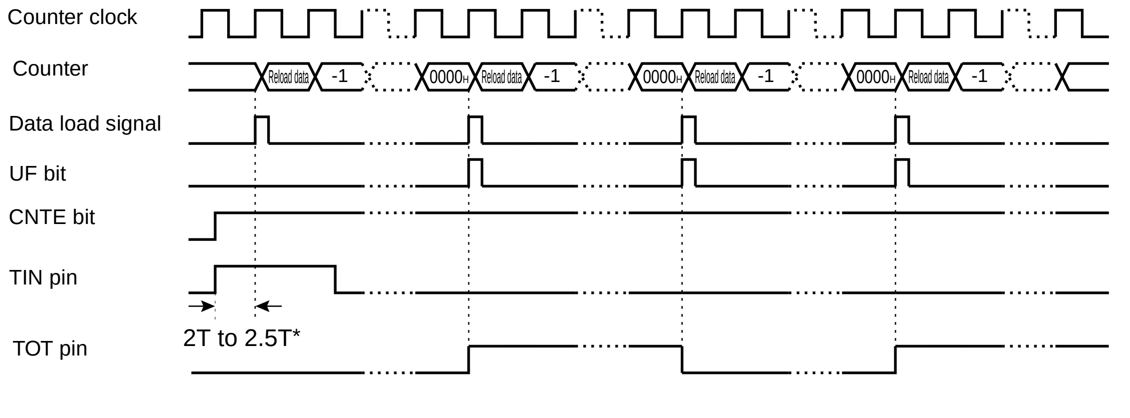
The trigger pulse width of the edge to be inputted to the TIN pin should be 2 T (T: machine cycles) or more.

Figure 14.5-6 Count Operation in External Trigger Mode (One-shot Mode)



T: Machine cycle
* : It takes 2T to 2.5T time to load data of reload register from external trigger input.

Figure 14.5-7 Count Operation in External Trigger Mode (Reload Mode)



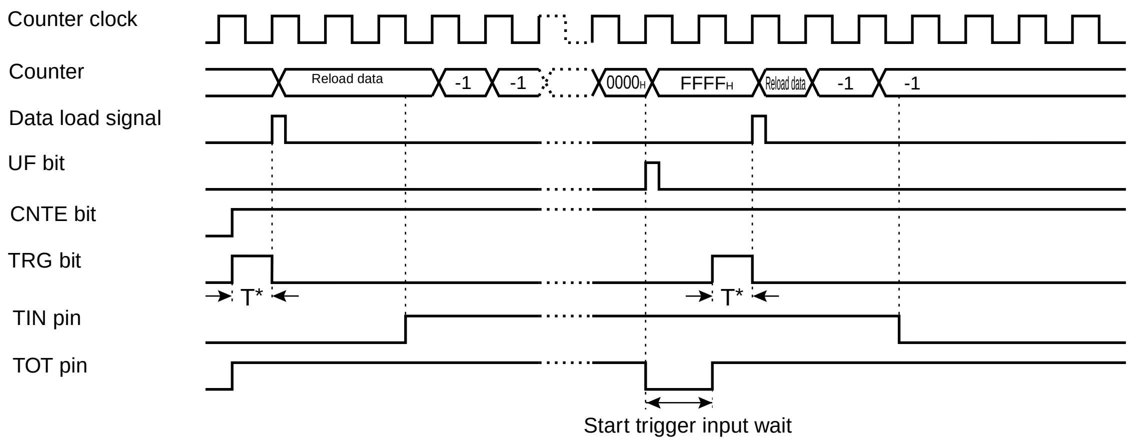
T : Machine cycle
* : It takes 2T to 2.5T time to load data of reload register from external trigger input.

[External gate input mode (MOD2 to MOD0="1x0$_B$", "1x1$_B$")]

When the external gate input mode is set, start the 16-bit reload timer by setting the software trigger bit in the timer control status register (TMCSR:TRG) to 1. When the 16-bit reload timer is started, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR).

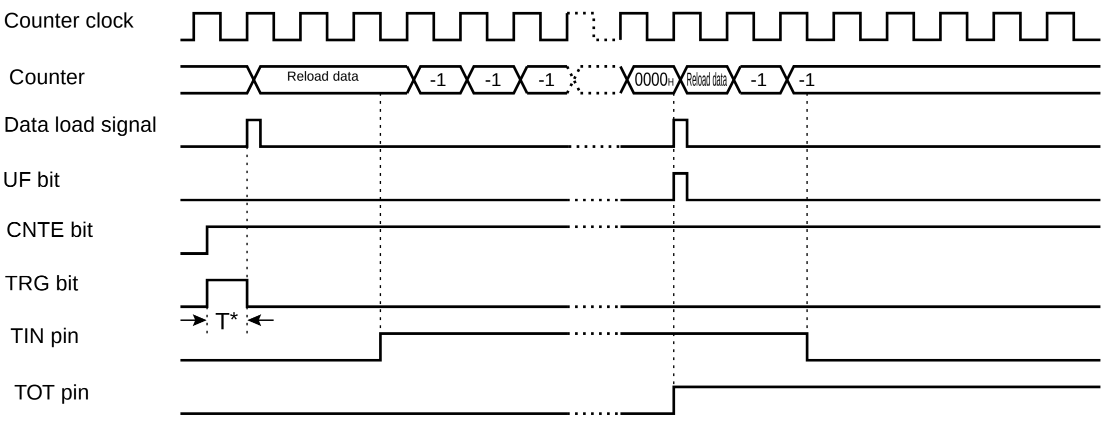
- After the 16-bit reload timer is started, the count operation of the TMR is performed while the set gate input level is inputted to the TIN pin.
- The gate input level (High or Low) can be selected by setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0).

Figure 14.5-8 Count Operation in External Gate Input Mode (One-shot Mode)



T : Machine cycle
* : It takes 1 T time to load data of reload register from trigger input.

Figure 14.5-9 Count Operation in External Gate Input Mode (Reload Mode)



T : Machine cycle
* : It takes 2T to 2.5T time to load data of reload register from trigger input.

14.5.2 Operation in Event Count Mode

In the event count mode, after the 16-bit reload timer is started, the edge of the signal input to the TIN pin is detected to perform the count operation of the 16-bit timer register (TMR). When the operation mode and the reload mode are set, a rectangular wave or a toggle wave is outputted from the TOT pin.

■ Setting of Event Count Mode

- The 16-bit reload timer is placed in the event count mode by setting the count clock select bits in the timer control status register (TMCSR:CSL1, CSL0) to "11_B".
- In the event count mode, the TMR decrements in synchronization with the edge detection of the external event clock input to the TIN pin.

[Setting initial value of counter]

After the 16-bit reload timer is started, the value set in the TMRLR is reloaded to the TMR.

1. Disables the operation of the 16-bit reload timer (TMCSR:CNTE = 0).
2. Sets a reload value to the TMR in the TMRLR.
3. Enables the operation of the 16-bit reload timer (TMCSR:CNTE = 1).

Note:

It takes 1 T (T: machine cycle) to load the value set in the TMRLR to the TMR after the start trigger is inputted.

■ Operation as 16-bit Timer Register Underflows

When the value of the 16-bit timer register (TMR) is decremented from "0000_H" to "$FFFF_H$" during the TMR count operation, an underflow occurs.

- When an underflow occurs, the underflow generating flag bit in the timer control status register (TMCSR:UF) is set to 1.
- When the underflow interrupt enable bit in the timer control status register (TMCSR:INTE) is set to 1, an underflow interrupt is generated.
- The reload operation when an underflow occurs is set by the reload select bit in the timer control status register (TMCSR:RELD).

[One-shot mode (TMCSR: RELD=0)]

When an underflow occurs, the TMR count operation is stopped, entering the start trigger input wait state. When the next start trigger is inputted, the TMR count operation is restarted.

In the one-shot mode, a rectangular wave is outputted from the TOT pin during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of the rectangular wave.

[Reload mode (TMCSR: RELD=1)]

When an underflow occurs, the value set in the TMRLR is reloaded to the TMR, continuing the TMR count operation.

In the reload mode, a toggle wave inverting the output level of the TOT pin is outputted each time an underflow occurs during the TMR count operation. The pin output level select bit in the timer control status register (TMCSR:OUTL) can be set to select the level (High or Low) of the toggle wave when the 16-bit reload timer is started.

■ Operation in Event Count Mode

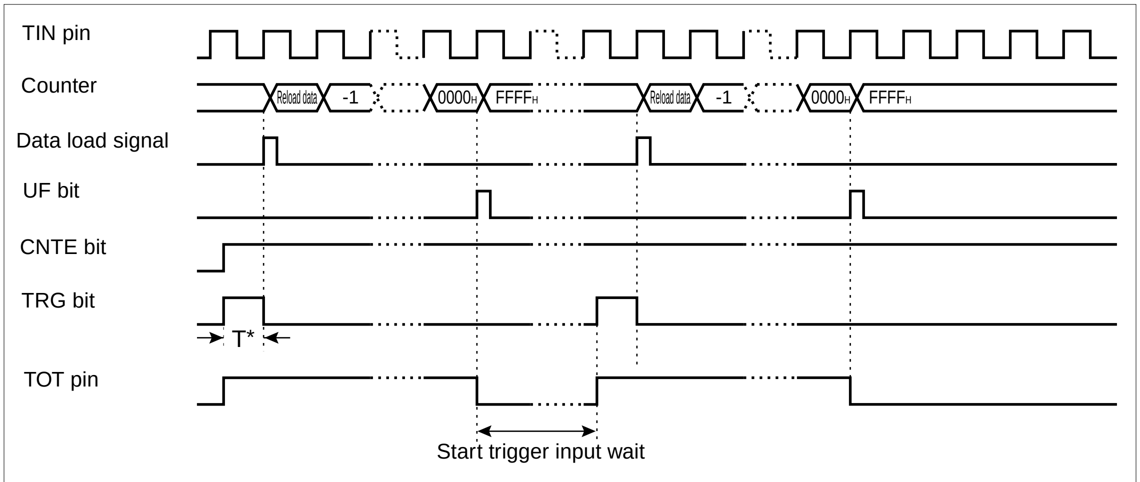
The operation of the 16-bit reload timer is enabled by setting the timer operation enable bit in the timer control status register (TMCSR:CNTE) to 1. When the software trigger bit in the timer control status register (TMCSR:TRG) is set to 1, the 16-bit reload timer is started. When the 16-bit reload timer is started, the value set in the 16-bit reload register (TMRLR) is reloaded to the 16-bit timer register (TMR), starting the TMR count operation. After the 16-bit reload timer is started, the edge of the external event clock input to the TIN pin is detected to perform the TMR count operation.

- By setting the operating mode select bits in the timer control status register (TMCSR:MOD2 to MOD0), the detected edge can be selected from the rising edge, falling edge, and both edges.

Note:

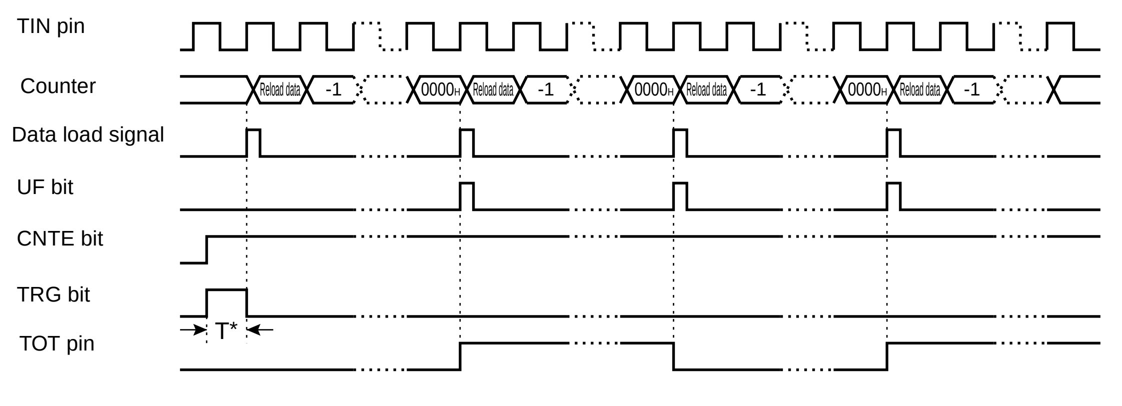
The level width of the external event clock to be inputted to the TIN pin should be 4 T (T: machine cycles) or more.

Figure 14.5-10 Count Operation in Event Count Mode (One-shot Mode)



T : Machine cycle
* : It takes 1 T time to load data of reload register from trigger input.

Figure 14.5-11 Count Operation in Event Count Mode (Reload Mode)



T : Machine cycle
* : It takes 1 T time to load data of reload register from trigger input.

14.6 Precautions when Using 16-bit Reload Timer

This section explains the precautions when using the 16-bit reload timer.

■ Precautions when Using 16-bit Reload Timer

● Precautions when setting by program

- Set the 16-bit reload register (TMRLR) after disabling the timer operation (TMCSR:CNTE = 0)
- The 16-bit timer register (TMR) can be read during the TMR count operation. However, always use the word instruction.
- Change the CSL1 and CSL0 bits in the TMCSR after disabling the timer operation (TMCSR:CNTE = 0)

● Precautions on interrupt

- When the UF bit in the TMCSR is set to 1 and the underflow interrupt output is enabled (TMCSR:INTE = 1), it is impossible to return from interrupt processing. Always clear the UF bit. However, when the EI^{2}OS is used, the UF bit is cleared automatically.
- When using the EI^{2}OS in the 16-bit reload timer, it is necessary to disable the interrupt of the 16-bit reload timer that shares the interrupt control register (ICR).

14.7 Sample Program of 16-bit Reload Timer

This section gives a program example of the 16-bit reload timer operated in the internal clock mode and the event count mode:

■ Program Example in Internal Clock Mode

● Processing specification

- The 24 ms interval timer interrupt is generated by the 16-bit reload timer 2.
- The repeated interrupts are generated in the reload mode.
- The timer is started using the software trigger instead of the external trigger input.
- EI^{2}OS is not used.
- The machine clock is 24 MHz; the count clock is 1.33 µs.

● Coding example

```
ICR04  EQU  0000B4H         ;Interrupt control register for 16-bit
                            ;reload timer
TMCSR2 EQU  000064H         ;Timer control status register
TMR2   EQU  00794CH         ;16-bit timer register
TMRLR2 EQU  00794CH         ;16-bit reload register
UF2    EQU  TMCSR2:2        ;Interrupt request flag bit
CNTE2  EQU  TMCSR2:1        ;Counter operation enable bit
TRG2   EQU  TMCSR2:0        ;Software trigger bit
;--------Main program----------------------------------
CODE   CSEG
;       :                   ;Stack pointer (SP), already initialized
       AND  CCR,#0BFH       ;Interrupts disabled
       MOV  I:ICR04,#00H    ;Interrupt level 0 (highest)
       CLRB I:CNTE2         ;Counter suspended
       MOVW I:TMRLR2,#4650H ;Set data for 24 ms timer
       MOVW I:TMCSR2,#0000100000011011B
                            ;Operation of interval timer,
                             clock = 1.33 µs
                            ;External trigger disabled, external
                             output disabled
                            ;Reload mode selected, interrupt enabled
                            ;Interrupt flag cleared, count started
       MOV  ILM,#07H        ;ILM in PS set to level 7
       OR   CCR,#40H        ;Interrupts enabled
LOOP:
        :
       Processing by user
        :
```

```
       BRA  LOOP             ;
;---------Interrupt program-----------------------------------
WARI:
       CLR  I:UF2            ;Interrupt request flag cleared
        :
        :
       Processing by user
        :
        :
       RETI                  ;Return from interrupt

CODE   ENDS
;---------Vector setting--------------------------------------
VECT   CSEG ABS=0FFH
       ORG  00FFB0H          ;Set vector to interrupt #19(13H)
       DSL  WARI
       ORG  00FFDCH          ;Reset vector set
       DSL  START
       DB   00H              ;Set to single-chip mode
VECT   ENDS
       END  START
```

■ Program Example in Event Counter Mode

● Processing specification

- An interrupt is generated when rising edges of the pulse input to the external event input pin are counted 10000 times by the 16-bit reload timer 2.
- Operation is performed in the one-shot mode.
- The rising edge is selected for the external trigger input.
- EI^{2}OS is not used.

● Coding example

```
ICR04  EQU  0000B4H         ;Interrupt control register for 16-bit
                            ;reload timer
TMCSR2 EQU  000064H         ;Timer control status register
TMR2   EQU  00794CH         ;16-bit timer register
TMRLR2 EQU  00794CH         ;16-bit reload register
DDR8   EQU  000018H         ;Port data register
UF2    EQU  TMCSR2:2        ;Interrupt request flag bit
CNTE2  EQU  TMCSR2:1        ;Counter operation enable bit
TRG2   EQU  TMCSR2:0        ;Software trigger bit
;---------Main program-----------------------------------
CODE   CSEG
;       :                   ;Stack pointer (SP), already initialized
                            ;Used at software starting mode(ACS1 :
                             STS1, 0 = 00B)
```

```
      AND   CCR,#0BFH      ;Interrupts disabled
      MOV   I:ICR04,#00H   ;Interrupt level 0 (highest)
      MOV   I:DDR8,00H     ;Sets P82/TIN2 pin to input
      CLRB  I:CNTE0        ;Counter suspended
      MOVW  I:TMRLR2,#2710H;Reload value set to 10000 times
      MOVW  I:TMCSR2,#0000110001001011B
                           ;Counter operation, rising edge,
                           ;and external output disabled
                           ;One-shot mode selected, interrupt enabled
                           ;Interrupt flag cleared, count started
      MOV   ILM,#07H       ;Set ILM in PS to level 7
      OR    CCR,#40H       ;Interrupts enabled
LOOP:
       :
      Processing by user
       :
      BRA   LOOP           ;
;---------Interrupt program-----------------------------------
WARI:
      CLR   I:UF2          ;Interrupt request flag cleared
       :
       :
      Processing by user
       :
       :
      RETI                 ;Return from interrupt

CODE  ENDS
;---------Vector setting----------------------------------------
VECT  CSEG  ABS=0FFH
      ORG   00FFB0H        ;Set vector to interrupt #19 (13H)
      DSL   WARI
      ORG   00FFDCH        ;Reset vector set
      DSL   START
      DB    00H            ;Set to single-chip mode
VECT  ENDS
      END   START
```

CHAPTER 15
WATCH TIMER

This chapter describes the functions and operations of the watch timer.

15.1 Overview of Watch Timer

The watch timer is a 15-bit free-run counter that increments in synchronization with the subclock.

- **Eight interval times can be selected and an interrupt request can be generated for each interval time.**
- **An operation clock can be supplied to the oscillation stabilization wait time timer of the subclock and the watchdog timer.**
- **The subclock is always used as a count clock regardless of the settings of the clock select register (CKSCR).**

■ Interval Timer Function

- When the watch timer reaches the interval time set by the interval time select bits (WTC:WTC2 to WTC0), the bit corresponding to the interval time of the watch timer counter overflows (carries) and the overflow flag bit is set (WTC:WTOF = 1).
- When the overflow flag bit is set (WTC:WTOF = 1) with interrupt enabled when an overflow occurs (WTC:WTIE = 1), an interrupt request is generated.
- The interval time of the watch timer can be selected from eight types shown in Table 15.1-1 .

Table 15.1-1 Interval Times of Watch Timer

Subclock Cycle	Interval Time
SCLK(122 µs)	2^8/SCLK(31.25 ms)
	2^9/SCLK(62.5 ms)
	2^{10}/SCLK(125 ms)
	2^{11}/SCLK(250 ms)
	2^{12}/SCLK(500 ms)
	2^{13}/SCLK(1.0 s)
	2^{14}/SCLK(2.0 s)
	2^{15}/SCLK(4.0 s)

SCLK: Subclock frequency
The parenthesized values are provided when the subclock operates at 8.192 kHz. Please consider the frequency difference of built-in CR oscillation when you use built-in CR oscillation clock as a sub-clock.

■ Cycle of Clock Supply

The watch timer supplies an operation clock to the oscillation stabilization wait time timer of the subclock and the watchdog timer. Table 15.1-2 shows the cycles of clocks supplied from the watch timer.

Table 15.1-2 Cycle of Clock Supplied from Watch Timer

Where to Supply Clock	Clock Cycle
Timer for oscillation stabilization wait time of subclock	2^{14}/SCLK(4.000 s)
Watchdog timer	2^{10}/SCLK(125 ms)
	2^{13}/SCLK(1.000 s)
	2^{14}/SCLK(2.000 s)
	2^{15}/SCLK(4.000 s)

SCLK: Subclock frequency
The parenthesized values are provided when the subclock operates at 8.192 kHz.

Note:

The frequency of the subclock (SCLK) is a value for 2 division/4 division of the clock inputted to the low-speed oscillation pin (X0A and X1A) or the internal CR oscillation clock.
The division ratio is set by the SCDS bit of the PLL/subclock control register (PSCCR).
When using the internal CR oscillation clock, see "CHAPTER 6 CLOCK SUPERVISOR". Please consider the frequency difference of built-in CR oscillation when you use built-in CR oscillation clock as a sub-clock.

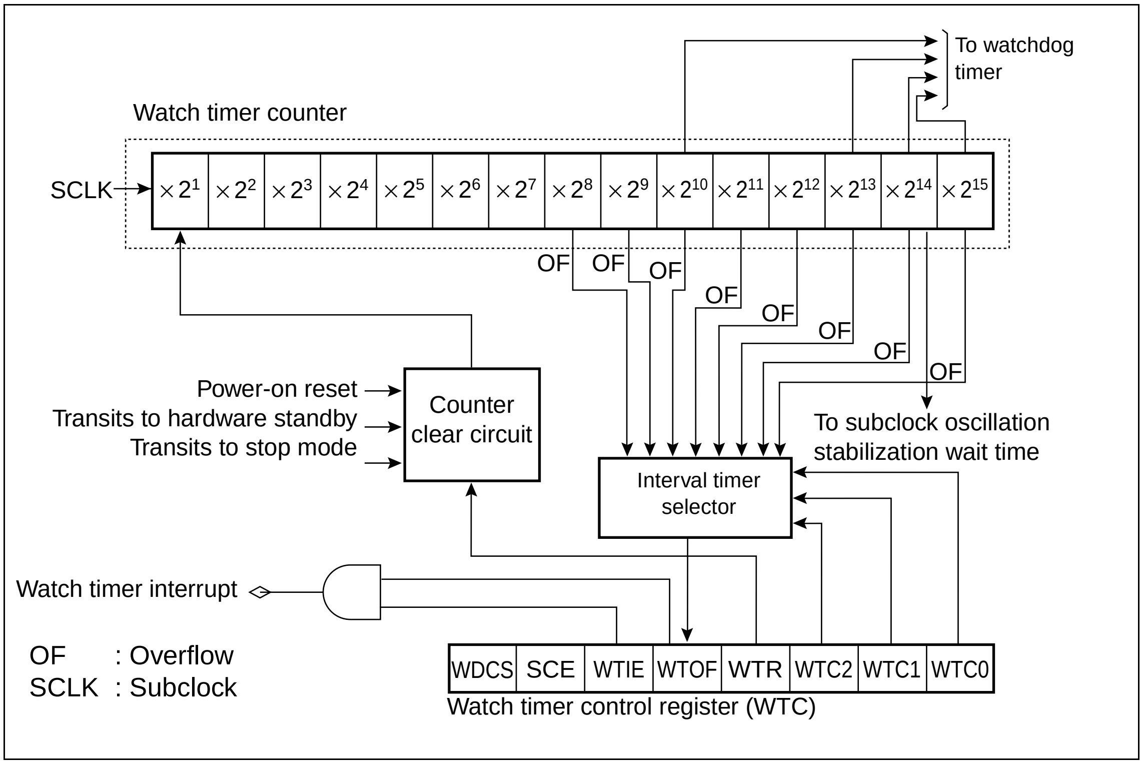
15.2 Block Diagram of Watch Timer

The watch timer consists of the following blocks:

- **Watch timer counter**
- **Counter clear circuit**
- **Interval timer selector**
- **Watch timer control register (WTC)**

■ Block Diagram of Watch Timer

Figure 15.2-1 Block Diagram of Watch Timer



The actual interrupt request number of the watch timer is as follows:

Interrupt request number: #27(1B$_H$)

● Watch timer counter

The watch timer counter is a 15-bit up counter that uses the subclock (SCLK) as a count clock.

● Counter clear circuit

The counter-clear circuit clears the watch timer counter.

● Interval timer selector

The interval timer selector sets the overflow flag bit when the watch timer counter reaches the interval time set in the watch timer control register (WTC).

● Watch timer control register (WTC)

The watch timer control register (WTC) selects the interval time, clears the watch timer counter, enables or disables an interrupt, checks the overflow (carry) state, and clears the overflow flag bit.

15.3 Configuration of Watch Timer

This section explains the registers and interrupt factors of the watch timer.

■ List of Registers and Reset Values of Watch Timer

Figure 15.3-1 List of Registers and Reset Values of Watch Timer

Watch timer control register (WTC)

Address: 0000AA$_H$

bit	7	6	5	4	3	2	1	0
	1	×	0	0	1	0	0	0

×: Undefined

■ Generation of Interrupt Request from Watch Timer

- When the interval time set by the interval time select bits (WTC:WTC2 to WTC0) is reached, the overflow flag bit (WTC:WTOF) is set to "1".
- When the overflow flag bit is set (WTC:WTOF = 1) with interrupt enabled when the watch timer counter overflows (carries) (WTC:WTIE = 1), an interrupt request is generated.

15.3.1 Watch Timer Control Register (WTC)

This section explains the functions of the watch timer control register (WTC).

■ Watch Timer Control Register (WTC)

Figure 15.3-2 Watch Timer Control Register (WTC)

Address	7	6	5	4	3	2	1	0	Reset value
0000AAH	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	$1X001000_B$
	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	

bit2 bit1 bit0

WTC2	WTC1	WTC0	Interval time select bit
0	0	0	2^{8}/SCLK(31.25 ms)
0	0	1	2^{9}/SCLK(62.5 ms)
0	1	0	2^{10}/SCLK(125 ms)
0	1	1	2^{11}/SCLK(250 ms)
1	0	0	2^{12}/SCLK(500 ms)
1	0	1	2^{13}/SCLK(1.0 s)
1	1	0	2^{14}/SCLK(2.0 s)
1	1	1	2^{15}/SCLK(4.0 s)

bit3

WTR	Watch timer clear bit	
	Read	Write
0	—	Clear watch timer counter
1	"1" is always read.	No effect

bit4

WTOF	Overflow flag bit	
	Read	Write
0	No overflow of the bit corresponding to set interval time	Clears WTOF bit
1	Overflow of the bit corresponding to set interval time	No effect

bit5

WTIE	Overflow interrupt enable bit
0	Interrupt request disable
1	Interrupt request enable

bit6

SCE	Oscillation stabilization wait time end bit
0	Oscillation stabilization wait state
1	Oscillation stabilization wait time end

bit7

WDCS	Watchdog clock select bit (input clock of watchdog timer)	
	Main or PLL clock mode	Subclock mode
0	Watch timer	Set "0"
1	Timebase timer	

R/W	: Read/Write
R	: Read only
X	: Undefined
SCLK	: Subclock
▨	: Reset value

The parenthesized values are provided when subclock operates at 8.192 kHz.

Table 15.3-1 Functions of Watch Timer Control Register (WTC)

Bit Name		Function
bit7	WDCS: Watchdog clock select bit	This bit selects the operation clock of the watchdog timer. **<Main clock mode or PLL clock mode>** **When set to "0":** Selects output of watch timer as operation clock of watchdog timer. **When set to "1":** Selects output of timebase timer as operation clock of watchdog timer. **<Subclock mode>** Always set this bit to 0 to select the output of the watch timer. Note: The watch timer and the timebase timer operate asynchronously. When the WDCS bit is changed from 0 to 1, the watchdog timer may run fast. The watchdog timer must be cleared before and after changing the WDCS bit.
bit6	SCE: Oscillation stabilization wait time end bit	This bit indicates that the oscillation stabilization wait time of the subclock ends. **When cleared to "0":** Subclock in oscillation stabilization wait state **When set to "1" :** Subclock oscillation stabilization wait time ends • The oscillation stabilization wait time of the subclock is fixed at 2^{14} /SCLK (SCLK: subclock frequency).
bit5	WTIE: Overflow interrupt enable bit	This bit enables or disables generation of an interrupt request when the watch timer counter overflows (carries). **When set to "0":** Interrupt request not generated even at overflow (WTOF=1) **When set to "1":** Interrupt request generated at overflow (WTOF = 1)
bit4	WTOF: Overflow flag bit	This bit is set to "1" when the counter value of the watch timer reaches the value set by the interval time select bit. When an overflow carry occurs (WTOF = 1) with interrupt request enabled (WTIE = 1), an interrupt request is generated. **When set to "0":** Clears watch timer counter **When set to "1":** No effect • The overflow flag bit is set to "1" when the bit of the watch timer counter corresponding to the interval time set by the interval time select bits (WTC2 to WTC0) overflows (carries).
bit3	WTR: Watch timer clear bit	This bit clears the watch timer counter. **When set to "0":** Clears watch timer counter to "0000_H". **When set to "1":** No effect **Read:** 1 is always read.
bit2 to bit0	WTC2, WTC1, WTC0: Interval time select bits	These bits set the interval time of the watch timer. • When the interval time set by the WTC2 to WTC0 bits is reached, the corresponding bit of the watch timer counter overflows (carries) and the overflow flag bit is set (WTC:WTOF = 1). • To set the WTC2 to WTC0 bits, set the WTOF bit to 0.

15.4 Watch Timer Interrupt

When the interval time is reached with the watch timer interrupt enabled, the overflow flag bit is set to "1" and an interrupt request is generated.

■ Watch Timer Interrupt

Table 15.4-1 shows the interrupt control bits and interrupt factors of the watch timer.

Table 15.4-1 Interrupt Control Bits of Watch Timer

	Watch Timer
Interrupt factor	Interval time of watch timer counter
Interrupt request flag bit	WTC: WTOF(overflow flag bit)
Interrupt factor enable bit	WTC: WTIE

- When the value set by the interval time select bits (WTC2 to WTC0) in the watch timer control register (WTC) is reached, the overflow flag bit in the WTC register is set to "1" (WTC:WTOF = 1).
- When the overflow flag bit is set (WTC:WTOF = 1) with the watch timer interrupt enabled (WTC:WTIE = 1), an interrupt request is generated.
- At interrupt processing, set the WTOF bit to 0 and cancel the interrupt request.

■ Watch Timer Interrupt and EI^{2}OS Transfer Function

- The watch timer does not correspond to the EI^{2}OS function.
- For details of the interrupt number, interrupt control register, and interrupt vector address, see "CHAPTER 3 INTERRUPTS".

15.5 Explanation of Operation of Watch Timer

The watch timer operates as an interval timer or an oscillation stabilization wait time timer of subclock. It also supplies an operation clock to the watchdog timer.

■ Watch Timer Counter

The watch timer counter continues incrementing in synchronization with the subclock (SCLK) while the subclock (SCLK) is operating.

● Clearing watch timer counter

The watch timer counter is cleared to "0000_H" when:

- A power-on reset occurs.
- The mode transits to the stop mode.
- The watch timer clear bit (WTR) in the watch timer control register (WTC) is set to "0".

Note:

When the watch timer counter is cleared, the interrupts of the watchdog timer and interval timer that use the output of the watch timer counter are affected.
To clear the watch timer by writing zero to the watch timer clear bit (WTR) in the watch timer control register (WTC), set the overflow interrupt enable bit (WTIE) in the WTC register to "0" and set the watch timer to interrupt inhibited state. Before permitting an interrupt, clear the interrupt request issued by writing zero to the overflow flag bit (WTOF) in the WTC register.

■ Interval Timer Function

The watch timer can be used as an interval timer by generating an interrupt at each interval time.

● Settings when using watch timer as interval timer

Operating the watch timer as an interval timer requires the settings shown in Figure 15.5-1 .

Figure 15.5-1 Setting of Watch Timer

	bit7	6	5	4	3	2	1	bit0
WTC	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0
	×	×	◎	◎	◎	◎	◎	◎

◎ : Used bit
× : Unused bit

- When the value set by the interval time select bits (WTC1, WTC0) in the watch timer control register (WTC) is reached, the overflow flag bit in the WTC register is set to "1" (WTC:WTOF = 1).
- When the overflow flag bit is set (WTC:WTOF = 1) with the overflow interrupt of the watch timer counter enabled (WTC:WTIE = 1), an interrupt request is generated.
- The overflow flag bit (WTC:WTOF) is set when the interval time is reached at the starting point of the timing at which the watch timer is finally cleared.

● Clearing overflow flag bit (WTC:WTOF)

When the mode is switched to the stop mode, the watch timer is used as an oscillation stabilization wait time timer of subclock. The WTOF bit is cleared concurrently with mode switching.

■ Setting Operation Clock of Watchdog Timer

The watchdog clock select bit (WDCS) in the watch timer control register (WTC) can be used to set the clock input source of the watchdog timer.

When using the subclock as the machine clock, always set the WDCS bit to 0 and select the output of the watch timer.

■ Oscillation Stabilization Wait Time Timer of Subclock

When the watch timer returns from the power-on reset and the stop mode, it functions as an oscillation stabilization wait time timer of subclock.

The subclock oscillation stabilization wait time is fixed at 2^{14} /SCLK (SCLK: subclock frequency).

Note:

Please consider the frequency difference of built-in CR oscillation when you use built-in CR oscillation clock as a sub-clock.

15.6 Program Example of Watch Timer

This section gives a program example of the watch timer.

■ Program Example of Watch Timer

- Processing specifications

An interval interrupt at 2^{13}/SCLK (SCLK: subclock) is generated repeatedly. The internal time is approximately 1.0 s (when subclock operates at 8.192 kHz).

- Coding example

```
ICR08  EQU   0000B8H             ;Interrupt control register
WTC    EQU   0000AAH             ;Watch timer control register
WTOF   EQU   WTC:4               ;Overflow flag bit
;
;---------Main program------------------------------------
CODE   CSEG
START:
;                                ;Stack pointer (SP) already
                                 ;initialized
       AND   CCR,#0BFH           ;Interrupt disabled
       MOV   I:ICR07,#00H        ;Interrupt level 0 (highest)
       MOV   I:WTC,#10100101B    ;Interrupt enabled
                                 ;Overflow flag cleared
                                 ;Watch timer counter cleared,
                                 ;$2^{13}$/SCLK(approx. 1.0 s)
       MOV   ILM,#07H            ;Set ILM in PS to level 7
       OR    CCR,#40H            ;Interrupt enabled
LOOP:
       •
       Processing by user
       •
       BRA   LOOP
;---------Interrupt program---------------------------------------
WARI:
       CLRB  I:WTOF              ;Overflow flag cleared
       •
       Processing by user
       •
       RETI                      ;Return from interrupt processing
CODE   ENDS
;---------Vector setting------------------------------------------
VECT   CSEG  ABS=0FFH
       ORG   00FF90H             ;Reset vector set #27 (1B$_H$)
       DSL   WARI
       ORG   00FFDCH            ;Reset vector set
       DSL   START
       DB    00H                ;Set to single-chip mode
VECT   ENDS
       END   START
```

CHAPTER 16

8-/16-BIT PPG TIMER

This chapter describes the functions and operations of the 8-/16-bit PPG timer.

16.1 Overview of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer is a reload timer module with two channels (PPGC and PPGD) that outputs a pulse in any cycle and at any duty ratio. A combination of two channels provides:

- **8-bit PPG output 2-channel independent operation mode**
- **16-bit PPG output operation mode**
- **8+8-bit PPG output operation mode**

The MB90360 series has two 8-/16-bit PPG timers. This section explains the functions of PPGC/D. PPGE/F has the same functions as PPGC/D.

■ Functions of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer consists of four 8-bit reload registers (PRLHC, PRLLC, PRLHD, and PRLLD) and two PPG down counters (PCNTC and PCNTD).

- Individual setting of High and Low widths in output pulse enables an output pulse of any cycle and duty ratio.
- The count clock can be selected from six internal clocks.
- The 8-/16-bit PPG timer can be used as an interval timer by generating an interrupt request at each interval time.
- An external circuit enables the 8-/16-bit PPG timer to be used as a D/A converter.

■ Operation Modes of 8-/16-bit PPG Timer

● 8-bit PPG output 2-channel independent operation mode

The 8-bit PPG output 2-channel independent operation mode causes the 2-channel modules (PPGC and PPGD) to operate as each independent 8-bit PPG timer.

Table 16.1-1 shows the interval times in the 8-bit PPG output 2-channel independent operation mode.

Table 16.1-1 Interval Times in 8-bit PPG Output 2-channel Independent Operation Mode

Count Clock Cycle	PPGC, PPGD	
	Interval Time	Output Pulse Time
1/ϕ(41.7 ns)	1/ϕ to 2^8/ϕ	2/ϕ to 2^9/ϕ
2/ϕ(83.3 ns)	2/ϕ to 2^9/ϕ	2^2/ϕ to 2^{10}/ϕ
2^2/ϕ(167 ns)	2^2/ϕ to 2^{10}/ϕ	2^3/ϕ to 2^{11}/ϕ
2^3/ϕ(333 ns)	2^3/ϕ to 2^{11}/ϕ	2^4/ϕ to 2^{12}/ϕ
2^4/ϕ(667 ns)	2^4/ϕ to 2^{12}/ϕ	2^5/ϕ to 2^{13}/ϕ
2^9/HCLK(128 µs)	2^9/HCLK to 2^{17}/HCLK	2^{10}/HCLK to 2^{18}/HCLK

HCLK: Oscillation clock
ϕ : Machine clock
The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock operates at 24 MHz.

● 16-bit PPG output operation mode

The 16-bit PPG output operation mode concatenates the 2-channel modules (PPGC and PPGD) to operate as a 16-bit 1-channel PPG timer.

Table 16.1-2 shows the interval times in this mode.

Table 16.1-2 Interval Times in 16-bit PPG Output Operation Mode

Count clock cycle	Interval time	Output pulse time
1/ϕ(41.7 ns)	1/ϕ to 2^{16}/ϕ	2/ϕ to 2^{17}/ϕ
2/ϕ(83.3 ns)	2/ϕ to 2^{17}/ϕ	2^2/ϕ to 2^{18}/ϕ
2^2/ϕ(167 ns)	2^2/ϕ to 2^{18}/ϕ	2^3/ϕ to 2^{19}/ϕ
2^3/ϕ(333 ns)	2^3/ϕ to 2^{19}/ϕ	2^4/ϕ to 2^{20}/ϕ
2^4/ϕ(667 ns)	2^4/ϕ to 2^{20}/ϕ	2^5/ϕ to 2^{21}/ϕ
2^9/HCLK(128 µs)	2^9/HCLK to 2^{25}/HCLK	2^{10}/HCLK to 2^{26}/HCLK

HCLK: Oscillation clock
ϕ : Machine clock
The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock operates at 24 MHz.

- 8+8-bit PPG output operation mode

The 8 + 8-bit PPG output operation mode causes the PPGC of the 2-channel modules to operate as an 8-bit prescaler and the underflow output of the PPGC to operate as the count clock of the PPGD.

Table 16.1-3 shows the interval times in this mode.

Table 16.1-3 Interval Times in 8+8-bit PPG Output Operation Mode

Count Clock Cycle	PPGC		PPGD	
	Interval Time	Output Pulse Time	Interval Time	Output Pulse Time
1/ϕ(41.7 ns)	1/ϕ to 2^{8}/ϕ	2/ϕ to 2^{9}/ϕ	1/ϕ to 2^{16}/ϕ	2/ϕ to 2^{17}/ϕ
2/ϕ(83.3 ns)	2/ϕ to 2^{9}/ϕ	2^{2}/ϕ to 2^{10}/ϕ	2/ϕ to 2^{17}/ϕ	2^{2}/ϕ to 2^{18}/ϕ
2^{2}/ϕ(167 ns)	2^{2}/ϕ to 2^{10}/ϕ	2^{3}/ϕ to 2^{11}/ϕ	2^{2}/ϕ to 2^{18}/ϕ	2^{3}/ϕ to 2^{19}/ϕ
2^{3}/ϕ(333 ns)	2^{3}/ϕ to 2^{11}/ϕ	2^{4}/ϕ to 2^{12}/ϕ	2^{3}/ϕ to 2^{19}/ϕ	2^{4}/ϕ to 2^{20}/ϕ
2^{4}/ϕ(667 ns)	2^{4}/ϕ to 2^{12}/ϕ	2^{5}/ϕ to 2^{13}/ϕ	2^{4}/ϕ to 2^{20}/ϕ	2^{5}/ϕ to 2^{21}/ϕ
2^{9}/HCLK(128µs)	2^{9}/HCLK to 2^{17}/HCLK	2^{10}/HCLK to 2^{18}/ HCLK	2^{9}/HCLK to 2^{25}/HCLK	2^{10}/HCLK to 2^{26}/ HCLK

HCLK: Oscillation clock
ϕ : Machine clock
The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock operates at 24 MHz.

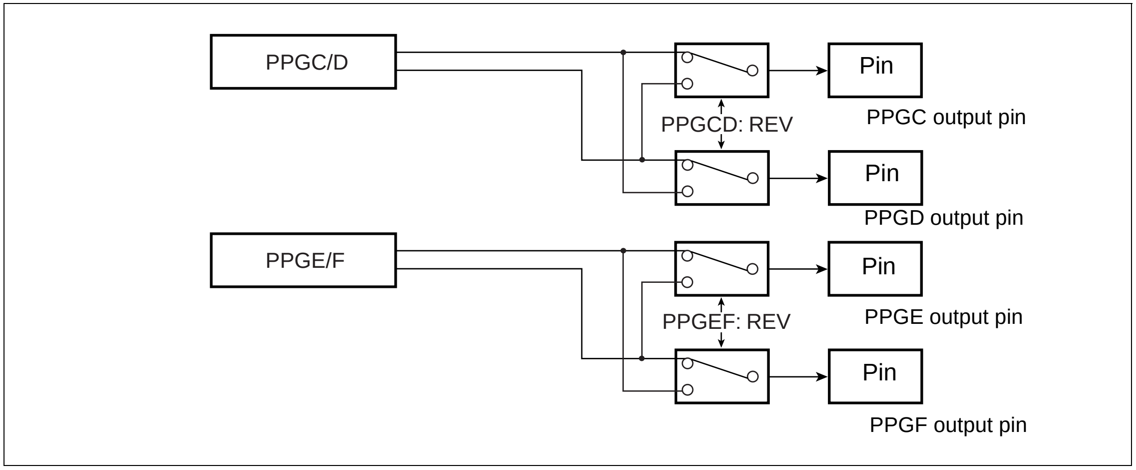
16.2 Block Diagram of 8-/16-bit PPG Timer

**The MB90360 series contains two 8-/16-bit PPG timers (each with 2 channels).
One 8-/16-bit PPG timer consists of 8-bit PPG timers with two channels.
This section shows the block diagrams for the 8-/16-bit PPG timer C and 8-/16-bit PPG timer D. The PPGE has the same function as the PPGC, and PPGF has the same function as PPGD.**

■ Channels and PPG Pins of PPG Timers

Figure 16.2-1 shows the relationship between the channels and the PPG pins of the 8-/16-bit PPG timers in the MB90360 series.

Figure 16.2-1 Channels and PPG Pins of PPG Timers

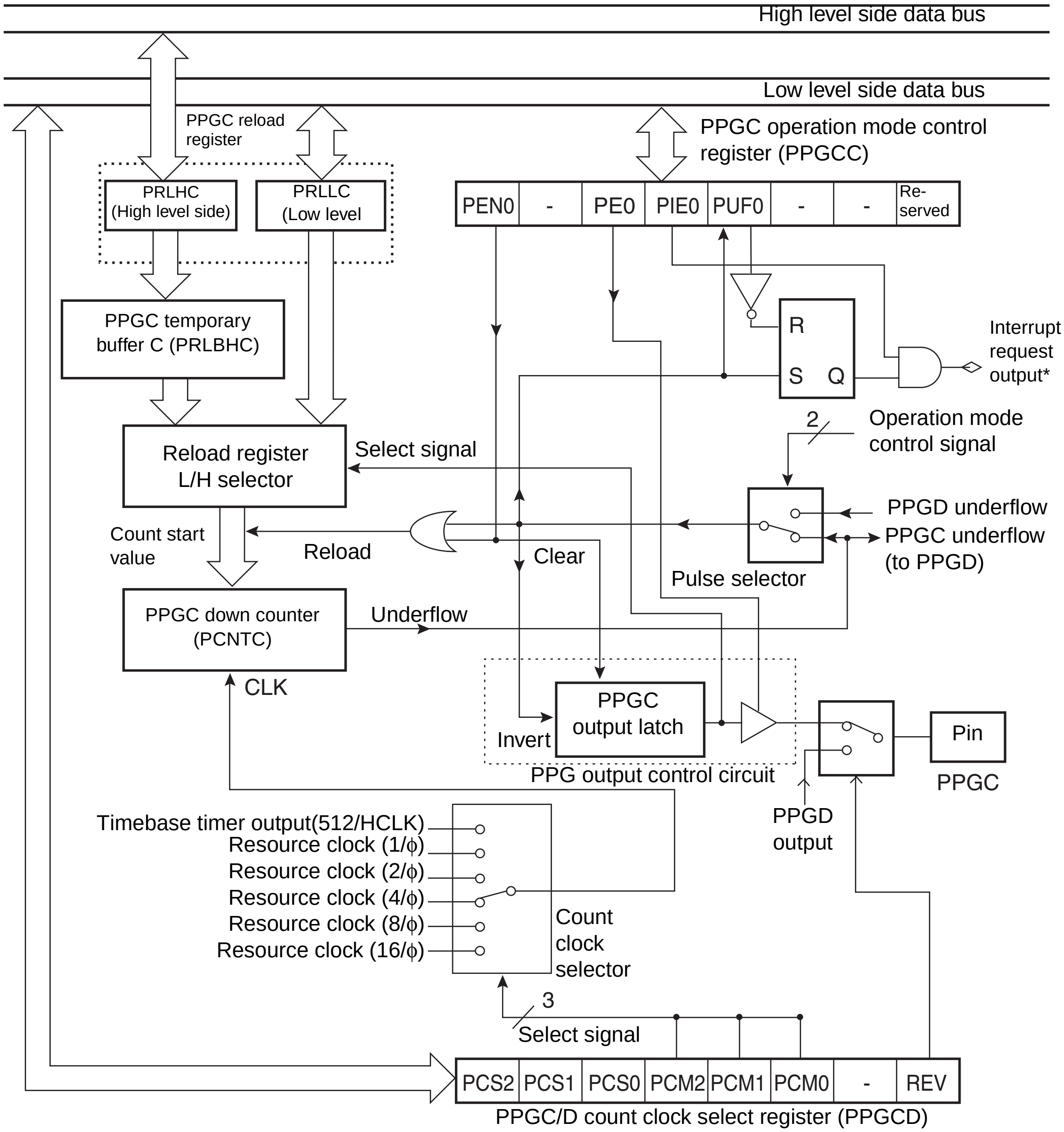


16.2.1 Block Diagram for 8-/16-bit PPG Timer C

The 8-/16-bit PPG timer C consists of the following blocks.

■ Block Diagram of 8-/16-bit PPG Timer C

Figure 16.2-2 Block Diagram of 8-/16-bit PPG Timer C



-	: Undefined
Reserved	: Reserved bit
HCLK	: Oscillation clock frequency
ϕ	: Machine clock frequency
*	: The interrupt output of 8-/16- bit PPG timer C is combined to one interrupt by OR circuit with the interrupt request output of PPG timer D.

- Details of pins in block diagram

Table 16.2-1 lists the actual pin names and interrupt request numbers of the 8-/16-bit PPG timer.

Table 16.2-1 Pins and Interrupt Request Numbers in Block Diagram

Channel	Output Pin		Interrupt Request Number
	PPG:REV=0	PPG:REV=1	
PPGC	P66 / PPGC	P22 / PPGD	#23 (17_H)
PPGD	P22 / PPGD	P66 / PPGC	
PPGE	P67/ PPGE	P23 / PPGF	#24 (18_H)
PPGF	P23/ PPGF	P67 / PPGE	

- PPG operation mode control register C (PPGCC)

This register enables or disables operation of the 8-/16-bit PPG timer, pin output and an underflow interrupt. It also indicates the occurrence of an underflow.

- PPGC/D count clock select register (PPGCD)

This register sets the count clock of the 8-/16-bit PPG timer and switching the output pin between PPGC and PPGD.

- PPGC reload registers (PRLHC, PRLLC)

These registers set the High width or Low width of the output pulse. The values set in these registers are reloaded to the PPGC down counter (PCNTC) when the 8-/16-bit PPG timer is started.

- PPGC down counter (PCNTC)

This counter is an 8-bit down counter that alternately reloads the values set in the PPGC reload registers (PRLHC and PRLLC) to decrement. When an underflow occurs, the pin output is inverted. The 2-channel PPG down counters (PPGC and PPGD) can also be concatenated for use as a single-channel 16-bit PPG down counter.

- PPGC temporary buffer (PRLBHC)

This buffer prevents deviation of the output pulse width caused at writing to the PPG reload registers (PRLHC and PRLLC). This buffer stores the PRLHC value temporarily and enables it in synchronization with the timing of writing to the PRLLC.

- Reload register L/H selector

This selector detects the current pin output level to select which register value, Low reload register (PRLLC) or High reload register (PRLHC), should be reloaded to the PPGC down counter.

- Count clock selector

This selector selects the count clock to be inputted to the PPGC down counter from five frequency-divided clocks of the machine clock or the frequency-divided clocks of the timebase timer.

- PPG output control circuit

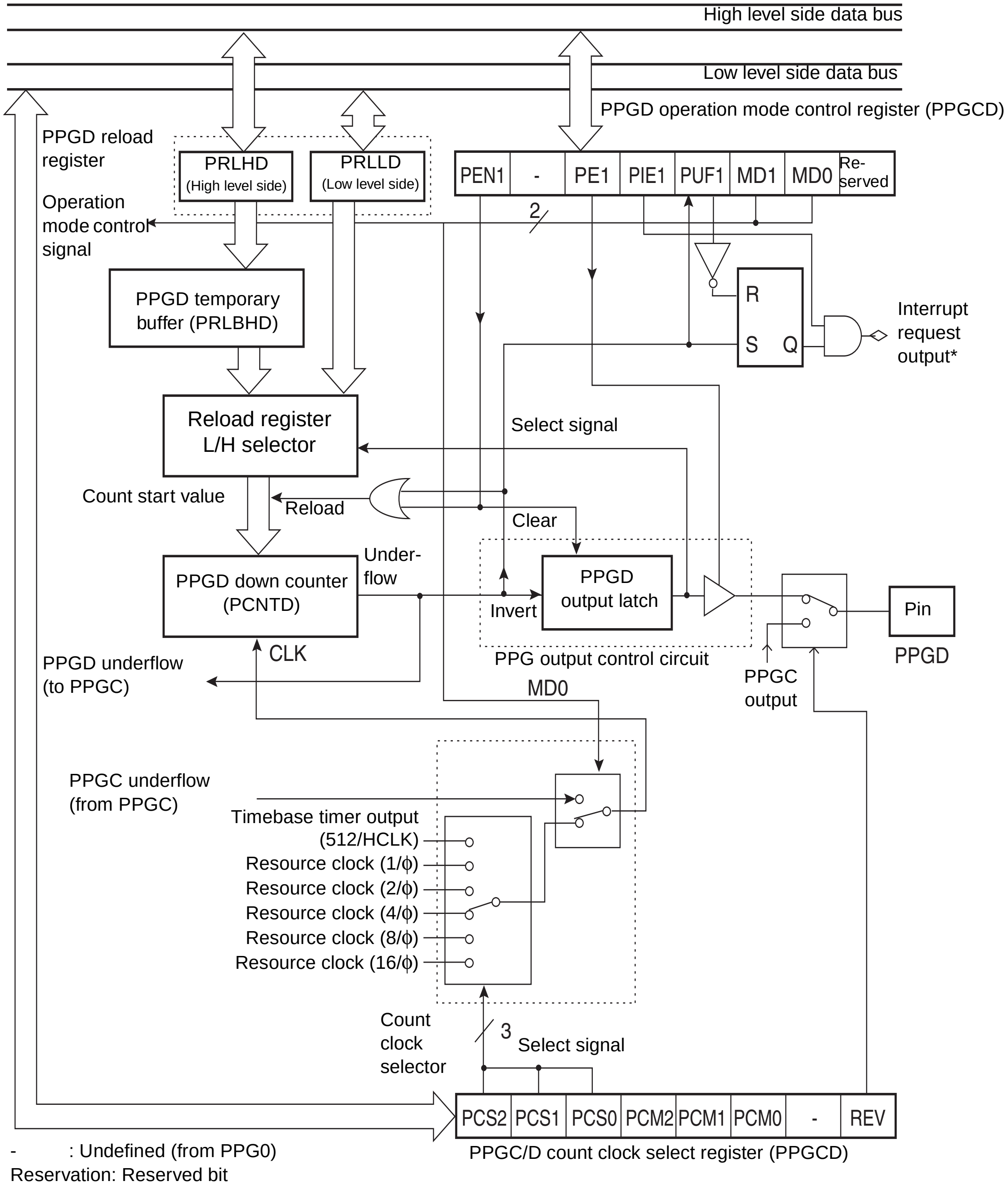
This circuit inverts the pin output level and the output when an underflow occurs.

16.2.2 Block Diagram of 8-/16-bit PPG Timer D

The 8-/16-bit PPG timer D consists of the following blocks.

■ Block Diagram of 8-/16-bit PPG Timer D

Figure 16.2-3 Block Diagram of 8-/16-bit PPG Timer D



-	: Undefined (from PPG0)
Reservation: Reserved bit	
HCLK	: Oscillation clock frequency
ϕ	: Machine clock frequency
*	: The interrupt output of 8-/16- bit PPG timer D is combined to one interrupt by OR circuit with the interrupt request output of PPG timer C.

- Details of pins in block diagram

Table 16.2-2 lists the actual pin names and interrupt request numbers of the 8-/16-bit PPG timer.

Table 16.2-2 Pins and Interrupt Request Numbers in Block Diagram

Channel	Output Pin		Interrupt Request Number
	PPG:REV=0	PPG:REV=1	
PPGC	P66 / PPGC	P22 / PPGD	#23 (17_H)
PPGD	P22 / PPGD	P66 / PPGC	
PPGE	P67/ PPGE	P23 / PPGF	#24 (18_H)
PPGF	P23/ PPGF	P67 / PPGE	

- PPG operation mode control register D (PPGCD)

This register sets the operation mode of the 8-/16-bit PPG timer, enables or disables the operation of the 8-/16-bit PPG timer D, the pin output and an underflow interrupt, and also indicates the generation of an underflow.

- PPGC/D count clock select register (PPGCD)

This register sets the count clock of the 8-/16-bit PPG timer.

- PPGD reload registers (PRLHD, PRLLD)

These registers set the High width or Low width of the output pulse. The values set in these registers are reloaded to the PPGD down counter (PCNTD) when the 8-/16-bit PPG timer D is started.

- PPGD down counter (PCNTD)

This counter is an 8-bit down counter that alternately reloads the values set in the PPGD reload registers (PRLHD and PRLLD) to decrement. When an underflow occurs, the pin output is inverted. The 2-channel PPG down counters (PPGC and PPGD) can also be connected for use as a single-channel 16-bit PPG down counter.

- PPGD temporary buffer (PRLBHD)

This buffer prevents deviation of the output pulse width caused at writing to the PPG reload registers (PRLHD and PRLLD). It stores the PRLHD value temporarily and enables it in synchronization with the timing of writing to the PRLLD.

- Reload register L/H selector

This selector detects the current pin output level to select which register value, Low reload register (PRLLD) or High reload register (PRLHD), should be reloaded to the PPGD down counter.

- Count clock selector

This selector selects the count clock to be inputted to the PPGD down counter from five frequency-divided clocks of the machine clock or the frequency-divided clocks of the timebase timer.

- PPG output control circuit

This circuit inverts the pin output level and the output when an underflow occurs.

16.3 Configuration of 8-/16-bit PPG Timer

This section explains the pins, registers and interrupt factors of the 8-/16-bit PPG timer.

■ Pins of 8-/16-bit PPG Timer

The pins of the 8-/16-bit PPG timer serve as general-purpose I/O ports. Table 16.3-1 indicates the pin functions and pin settings required to use the 8-/16-bit PPG timer.

Table 16.3-1 Pins of 8-/16-bit PPG Timer

Channel	Pin Name	Pin Function	Pin Setting Required for Use of 8-/16-bit PPG Timer
PPGC	P66 / AN6 / PPGC	General-purpose I/O port, A/D converter analog input 6/ PPG output C	• Analog input enable register : setting to disable(ADER6:ADE6=0) • PPG operating mode control register : pin output enable (PPGCC:PE0=1)
PPGD	P22 / PPGD	General-purpose I/O port, PPG output D	• PPG operating mode control register : pin output enable (PPGCD:PE1=1)
PPGE	P67 / AN7 / PPGE	General-purpose I/O port, A/D converter analog input 7/ PPG output E	• Analog input enable register : setting to disable (ADER6:ADE7=0) • PPG operating mode control register : pin output enable (PPGCE:PE0=1)
PPGF	P23 / PPGF	General-purpose I/O port, PPG output F	• PPG operating mode control register : pin output enable (PPGCF:PE1=1)

■ List of Registers and Reset Values of 8-/16-bit PPG Timer

Figure 16.3-1 List of Registers and Reset Values of 8-/16-bit PPG Timer

PPGm operation mode control register: H (PPGCm)	bit	15	14	13	12	11	10	9	8
		0	×	0	0	0	0	0	1
PPGn operation mode control register: L (PPGCn)	bit	7	6	5	4	3	2	1	0
		0	×	0	0	0	×	×	1
PPGn/m count clock select register (PPGnm)	bit	7	6	5	4	3	2	1	0
		0	0	0	0	0	0	×	0
PPGn reload register: H(PRLHn)	bit	15	14	13	12	11	10	9	8
		×	×	×	×	×	×	×	×
PPGn reload register: L(PRLLn)	bit	7	6	5	4	3	2	1	0
		×	×	×	×	×	×	×	×
PPGm reload register: H(PRLHm)	bit	15	14	13	12	11	10	9	8
		×	×	×	×	×	×	×	×
PPGm reload register: L(PRLLm)	bit	7	6	5	4	3	2	1	0
		×	×	×	×	×	×	×	×

×: Undefined

n = C, E
m = D, F

■ Generation of Interrupt Request from 8-/16-bit PPG Timer

In the 8-/16-bit PPG timer, the underflow generation flag bits in the PPG operation mode control registers (PPGCn:PUFn, PPGCm:PUFm) are set to "1" when an underflow occurs. If the underflow interrupts of channels causing an underflow are enabled (PPGCn: PIE0=1, PPGCm: PIE1=1), an underflow interrupt request is generated to the interrupt controller.

16.3.1 PPGC Operation Mode Control Register (PPGCC)

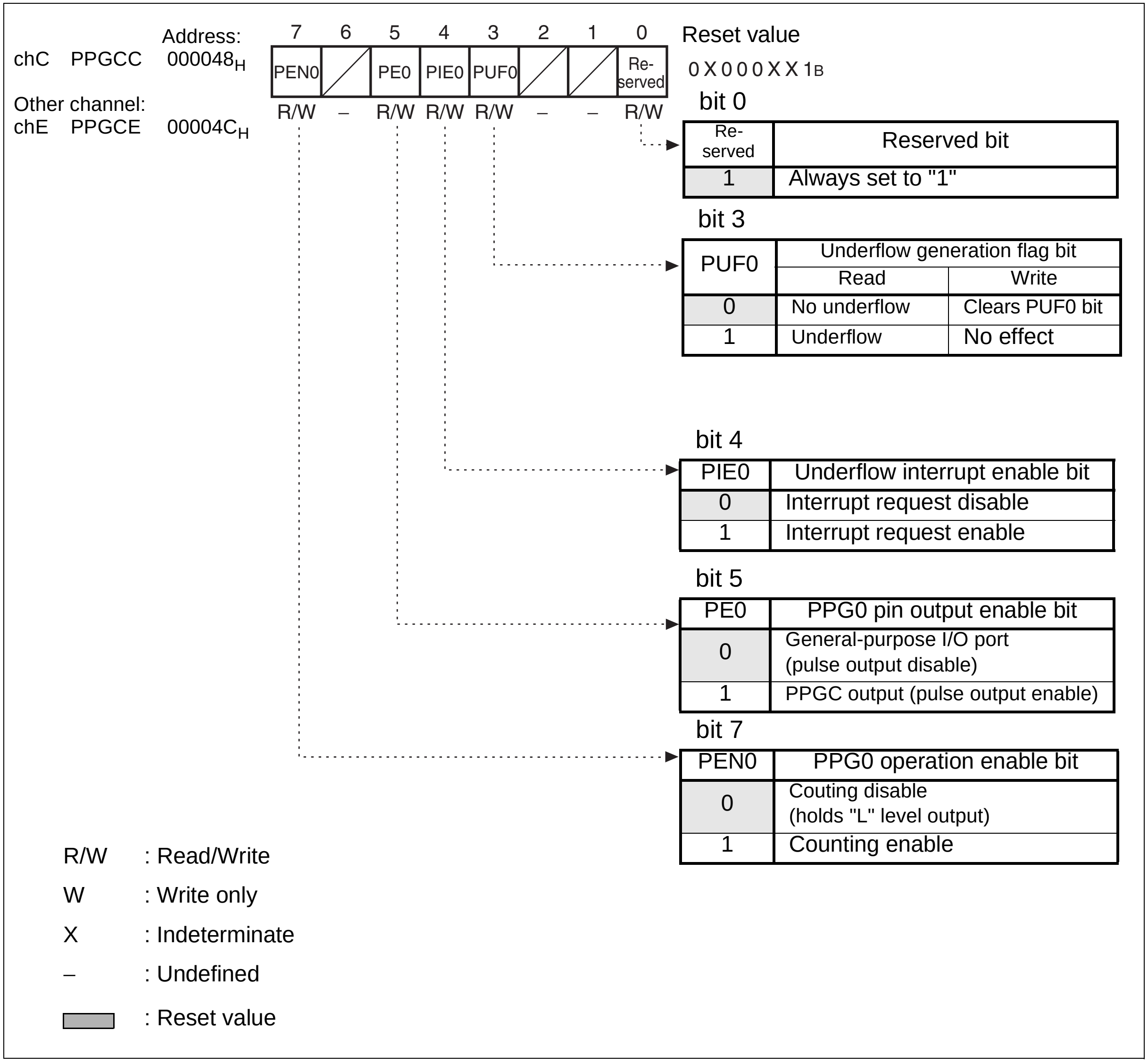
The PPGC operation mode control register (PPGC0) provides the following settings for the operation of 8-/16-bit PPG timer C:

- **Enabling or disabling operation of 8-/16-bit PPG timer C**
- **Switching between pin functions (enabling or disabling pulse output)**
- **Enabling or disabling underflow interrupt**
- **Setting underflow interrupt request flag**

This section explains the PPGCC function only. The PPGCE has the same function as the PPGCC, and the 8-/16-bit PPPG timer C and E is set.

■ PPGC Operation Mode Control Register (PPGCC)

Figure 16.3-2 PPGC Operation Mode Control Register (PPGCC)



bit 0

Re-served	Reserved bit
1	Always set to "1"

bit 3

PUF0	Underflow generation flag bit	
	Read	Write
0	No underflow	Clears PUF0 bit
1	Underflow	No effect

bit 4

PIE0	Underflow interrupt enable bit
0	Interrupt request disable
1	Interrupt request enable

bit 5

PE0	PPG0 pin output enable bit
0	General-purpose I/O port (pulse output disable)
1	PPGC output (pulse output enable)

bit 7

PEN0	PPG0 operation enable bit
0	Couting disable (holds "L" level output)
1	Counting enable

R/W	: Read/Write
W	: Write only
X	: Indeterminate
–	: Undefined
▨	: Reset value

Table 16.3-2 Functions of PPGC Operation Mode Control Register (PPGCC)

Bit Name		Function
bit7	PEN0: PPG0 operation enable bit	This bit enables or disables the count operation of the 8-/16-bit PPG timer C. **When set to "0":** Count operation disabled **When set to "1":** Count operation enabled • When the count operation is disabled (PEN0=0), and the pulse output is enabled (PE0=1), the output is held at a Low level.
bit6	Undefined bit	**Read:** The value is undefined. **Write:** No effect
bit5	PE0: PPG0 pin output enable bit	This bit switches between PPGC pin functions and enables or disables the pulse output. **When set to "0":** PPGC pin functions as general-purpose I/O port. The pulse output is disabled. **When set to "1":** PPGC pin functions as PPGC output pin. The pulse output is enabled.
bit4	PIE0: Underflow interrupt enable bit	This bit enables or disables an interrupt. **When set to "0":** No interrupt request generated even at underflow (PUF0 = 1). **When set to "1":** Interrupt request generated at underflow (PUF0 = 1)
bit3	PUF0: Underflow generation flag bit	**8-bit PPG output 2-channel independent operation mode, 8+8-bit PPG output operation mode:** When the value of the PPGC down counter is decremented from "00_H" to"FF_H", an underflow occurs (PUF0 = 1). **16-bit PPG output operation mode:** When the values of the PPGC and PPGD down counters are decremented from "0000_H" to "$FFFF_H$", an underflow occurs (PUF0 = 1). • When an underflow occurs (PUF0 = 1) with an underflow interrupt enabled (PIE0 = 1), an interrupt request is generated. **When set to "0":** Clears counter **When set to "1":** No effect **Read by read modify write instructions:** 1 is read.
bit2 bit1	Undefined bits	**Write:** No effect **Read:** The value is undefined.
bit0	Reserved: Reserved bit	Always set this bit to 1.

16.3.2 PPGD Operation Mode Control Register (PPGCD)

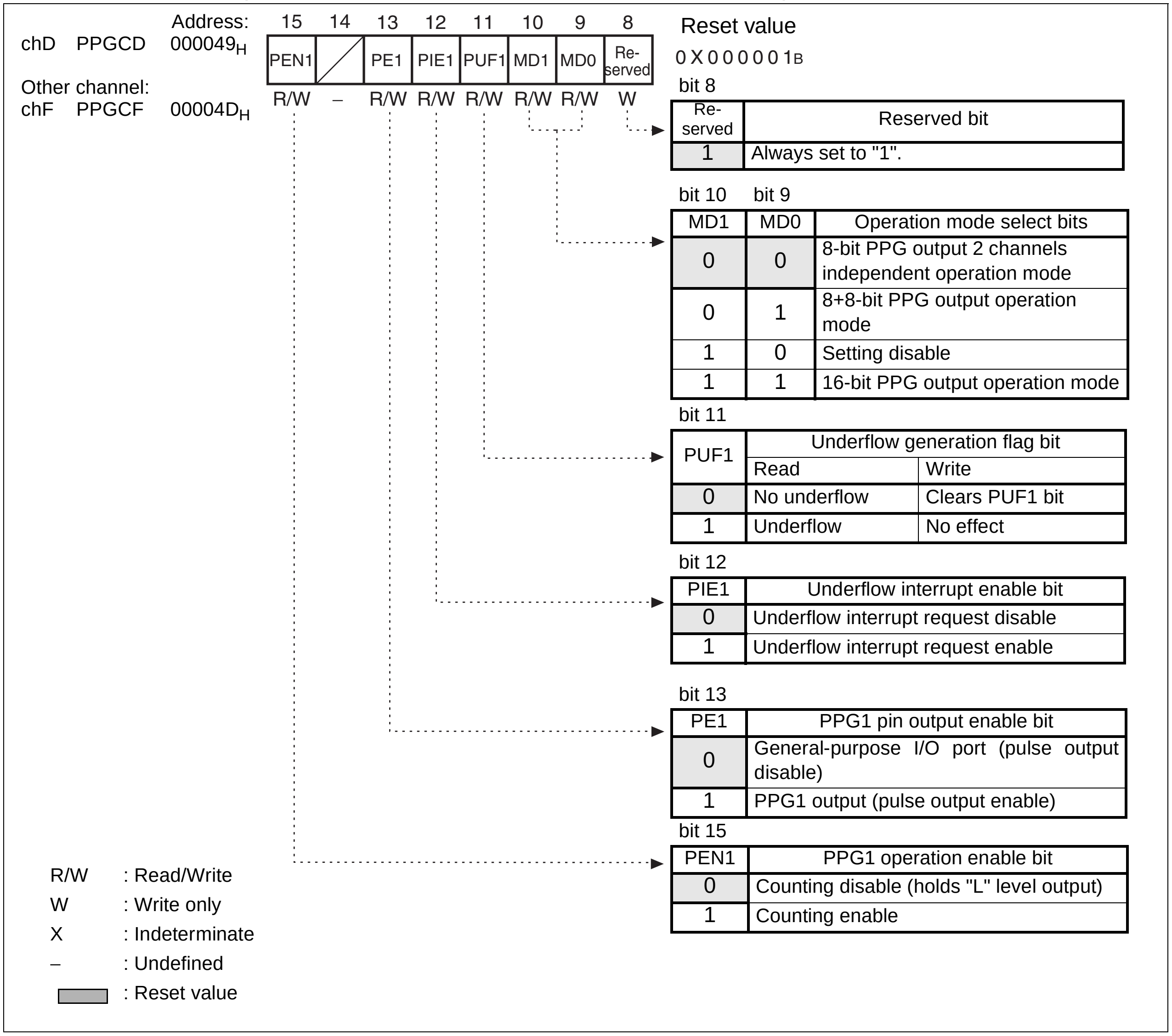
The PPGD operation mode control register provides the following settings about operation of 8-/16-bit PPG timer D:

- **Enabling or disabling operation of 8-/16-bit PPG timer D**
- **Switching between pin functions (enabling or disabling pulse output)**
- **Enabling or disabling underflow interrupt**
- **Setting underflow interrupt request flag**
- **Setting the operation mode of the 8-/16-bit PPG timer D and C**

This section explains the PPGCD function only. The PPGCF has the same function as the PPGCD, and the 8-/16-bit PPG timer F is set.

■ PPGD Operation Mode Control Register (PPGCD)

Figure 16.3-3 PPGD Operation Mode Control Register (PPGCD)



chD PPGCD Address: 000049$_H$

Other channel:
chF PPGCF 00004D$_H$

Address:	15	14	13	12	11	10	9	8
	PEN1		PE1	PIE1	PUF1	MD1	MD0	Re-served
	R/W	–	R/W	R/W	R/W	R/W	R/W	W

Reset value
0 X 0 0 0 0 0 1$_B$

bit 8

Re-served	Reserved bit
1	Always set to "1".

bit 10 bit 9

MD1	MD0	Operation mode select bits
0	0	8-bit PPG output 2 channels independent operation mode
0	1	8+8-bit PPG output operation mode
1	0	Setting disable
1	1	16-bit PPG output operation mode

bit 11

PUF1	Underflow generation flag bit	
	Read	Write
0	No underflow	Clears PUF1 bit
1	Underflow	No effect

bit 12

PIE1	Underflow interrupt enable bit
0	Underflow interrupt request disable
1	Underflow interrupt request enable

bit 13

PE1	PPG1 pin output enable bit
0	General-purpose I/O port (pulse output disable)
1	PPG1 output (pulse output enable)

bit 15

PEN1	PPG1 operation enable bit
0	Counting disable (holds "L" level output)
1	Counting enable

R/W	: Read/Write
W	: Write only
X	: Indeterminate
–	: Undefined
▨	: Reset value

Table 16.3-3 Functions of PPGD Operation Mode Control Register (PPGCD)

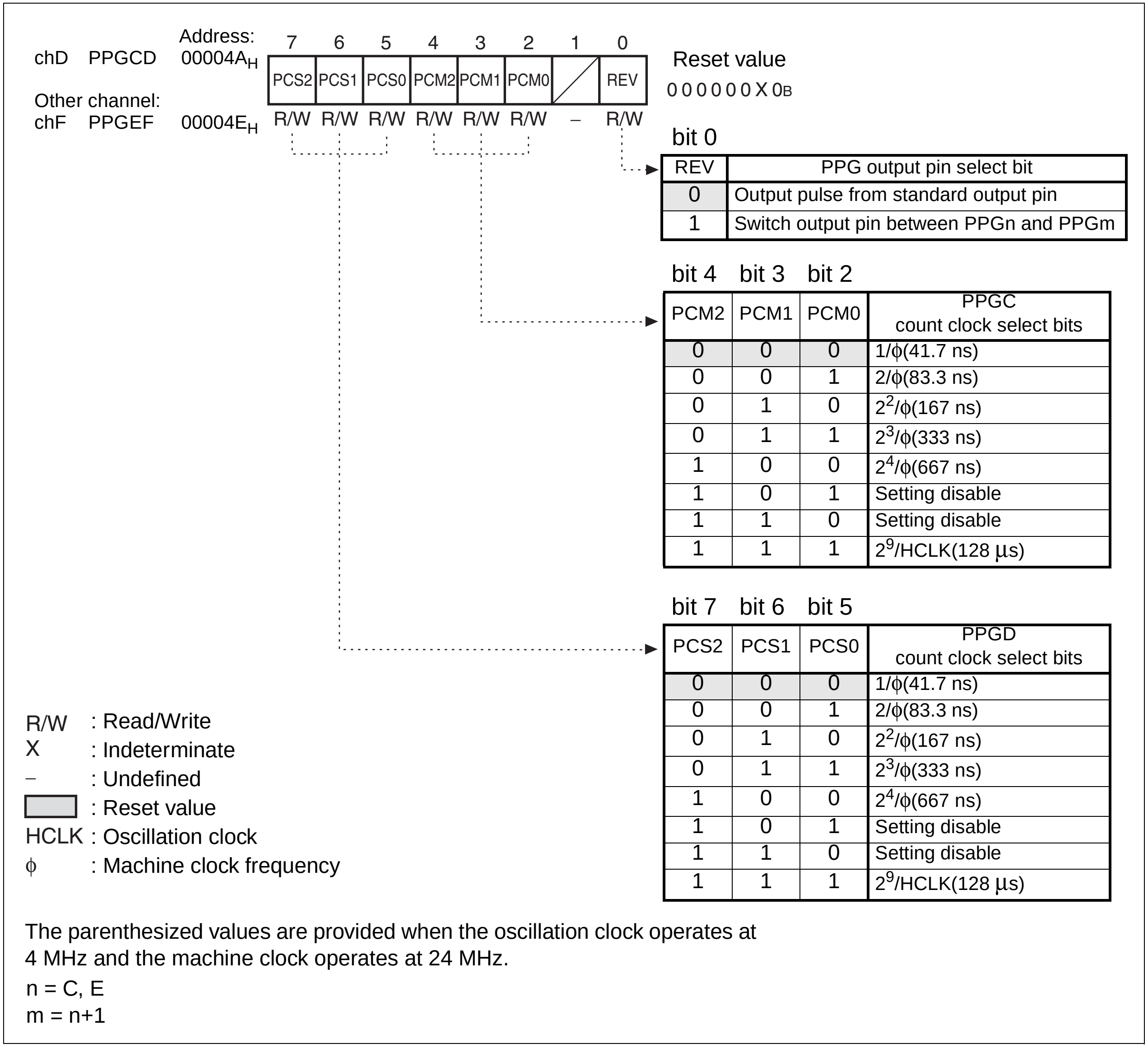
Bit name		Function
bit15	PEN1: PPG1 operation enable bit	This bit enables or disables the count operation of the 8-/16-bit PPG timer D. **When set to "0":** Count operation disabled **When set to "1":** Count operation enabled • When the count operation is disabled (PEN1 = 0), and the pulse output is enabled (PE1=1), the output is held at a Low level.
bit14	Undefined bit	**Read:** The value is undefined. **Write:** No effect
bit13	PE1: PPG1 Pin output enable bit	This bit switches between PPGD pin functions and enables or disables the pulse output. **When set to "0":** PPGD pin functions as general-purpose I/O port. The pulse output is disabled. **When set to "1":** PPGD pin functions as PPGD output pin. The pulse output is enabled.
bit12	PIE1: Underflow interrupt enable bit	This bit enables or disables an interrupt. **When set to "0":** No interrupt request is generated even at underflow (PUF1 = 1) **When set to "1":** Interrupt request is generated at underflow (PUF1 = 1)
bit11	PUF1: Underflow generation flag bit	**8-bit PPG output 2-channel independent operation mode, 8+8-bit PPG output operation mode:** When the value of the PPGD down counter is decremented from "00_H" to "FF_H", an underflow occurs (PUF1 = 1). **16-bit PPG output operation mode:** When the values of the PPGC and PPGD down counters are decremented from "0000_H" to "$FFFF_H$", an underflow occurs (PUF1 = 1). • When an underflow occurs (PUF1 = 1) with an underflow interrupt request enabled (PIE1 = 1), an interrupt request is generated. **When set to "0":** Clears counter **When set to "1":** No effect **Read by read modify write instructions:** 1 is read.
bit10 bit9	MD1, MD0: Operation mode select bits	These bits set the operation mode of the 8-/16-bit PPG timer. **[Any mode other than 8-bit PPG output 2-channel independent operation mode]** • Use a word instruction to set the PPG operation enable bits (PEN0 and PEN1) at one time. • Do not set operation of only one of the two channels (PEN1 = 0/PEN0 = 1 or PEN1 = 1/ PEN0 = 0). Note: Do not set the MD1 and MD0 bits to "10_B".
bit8	Reserved: Reserved bit	Always set this bit to 1.

16.3.3 PPGC/D Count Clock Select Register (PPGCD)

The PPGC/D count clock select register selects the count clock of the 8-/16-bit PPG timers C and D and the output pin.
This section explains the PPGCD function only. The PPGEF has the same function as the PPGCD, and the 8-/16-bit PPG timers E and F are set.

■ PPGC/D Count Clock Select Register (PPGCD)

Figure 16.3-4 PPGC/D Count Clock Select Register (PPGCD)



chD PPGCD Address: 00004A$_H$
Other channel:
chF PPGEF 00004E$_H$

	7	6	5	4	3	2	1	0
	PCS2	PCS1	PCS0	PCM2	PCM1	PCM0		REV
	R/W	R/W	R/W	R/W	R/W	R/W	–	R/W

Reset value
0 0 0 0 0 0 X 0B

bit 0

REV	PPG output pin select bit
0	Output pulse from standard output pin
1	Switch output pin between PPGn and PPGm

bit 4 PCM2	bit 3 PCM1	bit 2 PCM0	PPGC count clock select bits
0	0	0	1/ϕ(41.7 ns)
0	0	1	2/ϕ(83.3 ns)
0	1	0	2^2/ϕ(167 ns)
0	1	1	2^3/ϕ(333 ns)
1	0	0	2^4/ϕ(667 ns)
1	0	1	Setting disable
1	1	0	Setting disable
1	1	1	2^9/HCLK(128 µs)

bit 7 PCS2	bit 6 PCS1	bit 5 PCS0	PPGD count clock select bits
0	0	0	1/ϕ(41.7 ns)
0	0	1	2/ϕ(83.3 ns)
0	1	0	2^2/ϕ(167 ns)
0	1	1	2^3/ϕ(333 ns)
1	0	0	2^4/ϕ(667 ns)
1	0	1	Setting disable
1	1	0	Setting disable
1	1	1	2^9/HCLK(128 µs)

R/W	: Read/Write
X	: Indeterminate
–	: Undefined
▨	: Reset value
HCLK	: Oscillation clock
ϕ	: Machine clock frequency

The parenthesized values are provided when the oscillation clock operates at 4 MHz and the machine clock operates at 24 MHz.

n = C, E
m = n+1

Table 16.3-4 Functions of PPGC/D Count Clock Select Register (PPGCD)

Bit Name		Function
bit7 to bit5	PCS2 to PCS0: PPGD count clock select bits	These bits set the count clock of the 8-/16-bit PPG timer D. • The count clock can be selected from five frequency-divided clocks of the machine clock and the frequency-divided clocks of the timebase timer. • The settings of the PPGD count clock select bits (PCS2 to PCS0) are enabled only in the 8-bit PPG output 2-channel independent mode (PPGCD: MD1, MD0="00_B").
bit4 to bit2	PCM2 to PCM0: PPGC count clock select bits	These bits set the count clock of the 8-/16-bit PPG timer C. • The count clock can be selected from five frequency-divided clocks of the machine clock and the frequency-divided clocks of the timebase timer.
bit1	Undefined bit	**Read:** The value is undefined. **Write:** No effect
bit0	REV: PPG output pin select bit	This bit switches the output pin in the 8-/16-bit PPG timer C and D. **When set to "0":** Output from the standard output pin. PPGC → PPGC output pin PPGD → PPGD output pin **When set to "1":** Switch the output pin. PPGC → PPGD output pin PPGD → PPGC output pin

16.3.4 PPG Reload Registers (PRLLC/PRLHC, PRLLD/PRLHD)

The value (reload value) from which the PPG down counter starts counting is set in the PPG reload registers, which are an 8-bit register at Low level and an 8-bit register at High level.
This section explains the function of PRLLC/PRLHC and PRLLD/PRLHD only. The PRLLE/PRLHE, PRLLF/PRLHF have the same function as the PRLLC/PRLHC, and the 8-/16-bit PPG timers E, F are set.

■ PPG Reload Registers (PRLLC/PRLHC, PRLLD/PRLHD)

Figure 16.3-5 PPG Reload Registers (PRLLC/PRLHC, PRLLD/PRLHD)

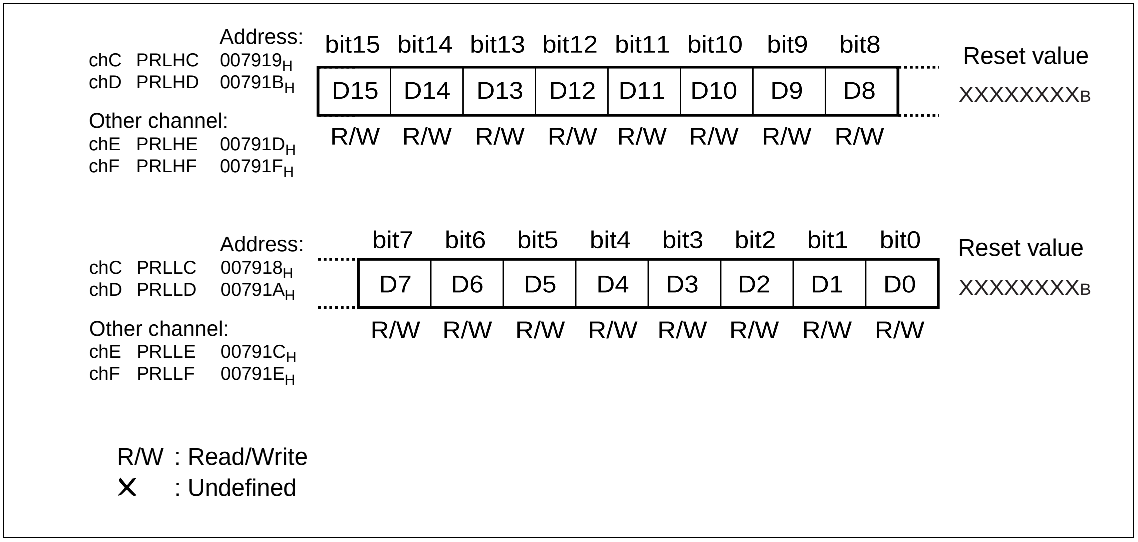


Table 16.3-5 indicates the functions of the PPG reload registers.

Table 16.3-5 Functions of PPG Reload Registers

Function	8-/16-bit PPG Timer C	8-/16-bit PPG Timer D
Retains reload value on Low-level side	PRLLC	PRLLD
Retains reload value on High-level side	PRLHC	PRLHD

Notes:

- In the 16-bit PPG output operation mode (PPGCD: MD1, MD0="11$_B$"), use a long-word instruction to set the PPG reload registers or the word instruction to set the PPGC and PPGD in this order.
- In the 8 + 8-bit PPG output operation mode (PPGCD: MD1, MD0="01$_B$"), set the same value in both the Low-level and High-level PPG reload registers (PRLLC/PRLHC) of the 8-/16-bit PPG timer C. Setting a different value in the Low-level and High-level PPG reload registers may cause the 8-/16-bit PPG timer D to have different PPG output waveforms at each clock cycle.

16.4 Interrupts of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer can generate an interrupt request when the PPG down counter underflows. It also not corresponds to the EI^{2}OS.

■ Interrupts of 8-/16-bit PPG Timer

Table 16.4-1 shows the interrupt control bits and interrupt factor of the 8-/16-bit PPG timer.

Table 16.4-1 Interrupt Control Bits of 8-/16-bit PPG Timer

	PPGn	PPGm
Interrupt request flag bit	PPGCn: PUF0	PPGCm: PUF1
Interrupt request enable bit	PPGCn: PIE0	PPGCm: PIE1
Interrupt factor	Underflow in PPGn down counter	Underflow in PPGm down counter

Note: n = C, E   m = n + 1

[8-bit PPG output 2-channel independent operation mode or 8 + 8-bit PPG output operation mode]

- In the 8-bit PPG output 2-channel independent operation mode or the 8 + 8-bit PPG output operation mode, the PPGn and PPGm timers can generate an interrupt independently.
- When the value of the PPGn or PPGm down counter is decremented from "00_H" to "FF_H", an underflow occurs. When an underflow occurs, the underflow generation flag bit in the channel causing an underflow is set (PPGCn: PUF0=1 or PPGCm: PUF1=1).
- If an interrupt request from the channel that causes an underflow is enabled (PPGCn: PIE0=1 or PPGCm: PIE1=1), an interrupt request is generated.

[16-bit PPG output operation mode]

- In the 16-bit PPG output operation mode, when the values of the PPGn and PPGm down counters are decremented from "0000_H" to "$FFFF_H$", an underflow occurs. When an underflow occurs, the underflow generation flag bits in the two channels are set at one time (PPGCn: PUF0=1 or PPGCm: PUF1=1).
- When an underflow occurs with either of the two channel of the interrupt requests enabled (PPGCn: PIE1=0, PPGCm: PIE1=1 or PPGCn: PIE0=1, PPGCm: PIE0=0), an interrupt request is generated.
- To prevent duplication of interrupt requests, disable either of the two channel of the underflow interrupt enable bits in advance (PPGCn: PIE0=0, PPGCm: PIE1=1 or PPGCn: PIE0=1, PPGCm: PIE1=0).
- When the two channels of the underflow generation flag bits are set (PPGCn: PUF0=1 and PPGCm: PUF1=1), clear the two channels at the same time.

■ Interrupt of 8-/16-bit PPG Timer

For details of the interrupt number, interrupt control register, and interrupt vector address, see "CHAPTER 3 INTERRUPTS".

16.5 Explanation of Operation of 8-/16-bit PPG Timer

The 8-/16-bit PPG timer outputs a pulse width at any frequency and at any duty ratio continuously.

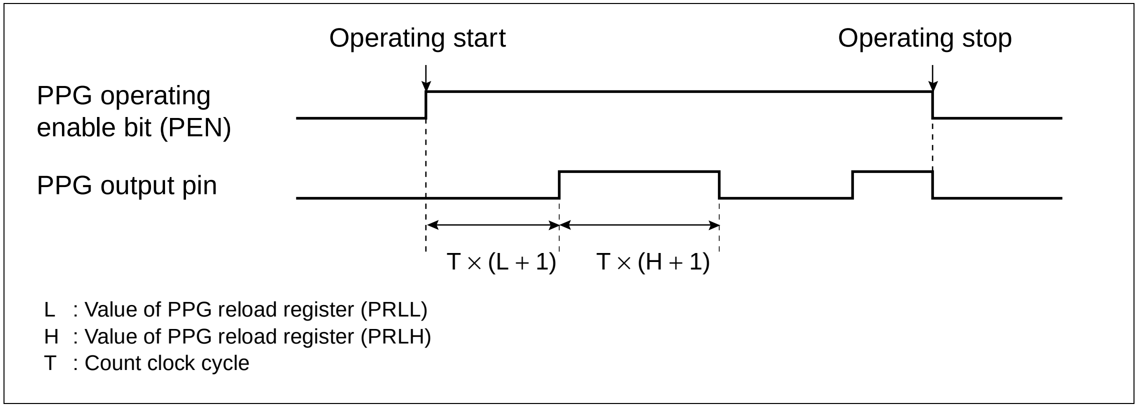
■ Operation of 8-/16-bit PPG Timer

● Output operation of 8-/16-bit PPG timer

- The 8-/16-bit PPG timer has two (Low-level and High-level) 8-bit reload registers (PRLLn/PRLHn and PRLLm/PRLHm) for per channel.
- The values set in the 8-bit reload registers (PRLLn/PRLHn and PRLLm/PRLHm) are reloaded alternately to the PPG down counters (PCNTn and PCNTm).
- After reloading the values in the PPG down counters, decrementing is performed in synchronization with the count clocks set by the PPG count clock select bits (PPGnm: PCM2 to PCM0 and PCS2 to PCS0).
- If the values set in the reload registers are reloaded to the PPG down counters when an underflow occurs, the pin output is inverted.

Figure 16.5-1 shows the output waveform of the 8-/16-bit PPG timer.

Figure 16.5-1 Output Waveform of 8-/16-bit PPG Timer


Operating start
Operating stop
PPG operating enable bit (PEN)
PPG output pin
T × (L + 1)
T × (H + 1)
L : Value of PPG reload register (PRLL)
H : Value of PPG reload register (PRLH)
T : Count clock cycle

● Operation modes of 8-/16-bit PPG timer

As long as the operation of the 8-/16-bit PPG timer is enabled (PPGCn: PEN0=1, PPGCm: PEN1=1), a pulse waveform is outputted continuously from the PPG output pin. A pulse width of any frequency and duty ratio can be set.

The pulse output of the 8-/16-bit PPG timer is not stopped until operation of the 8-/16-bit PPG timer is stopped (PPGCn: PEN0=0, PPGCm: PEN1=0).

- 8-bit PPG output 2-channel independent operation mode
- 16-bit PPG output operation mode
- 8 + 8-bit PPG output operation mode

Note: n = C, E
m = n+1

16.5.1 8-bit PPG Output 2-channel Independent Operation Mode

In the 8-bit PPG output 2-channel independent operation mode, the 8-/16-bit PPG timer is set as an 8-bit PPG timer with two independent channels. PPG output operation and interrupt request generation can be performed independently for each channel.

■ Setting for 8-bit PPG Output 2-channel Independent Operation Mode

Operating the 8-/16-bit PPG timer in the 8-bit PPG output 2-channel independent operation mode requires the setting shown in Figure 16.5-2 .

Figure 16.5-2 Setting for 8-bit PPG Output 2-channel Independent Operation Mode

	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
PPGCm/PPGCn	PEN1	–	PE1	PIE1	PUF1	MD1	MD0	Re-served	PEN0	–	PE0	PIE0	PUF0	–	–	Re-served
	1		◎	◎	◎	0	0	1	1		◎	◎	◎			1
PPGnm	(Reserved area)								PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	–	REV
									◎	◎	◎	◎	◎	◎		◎

PRLHn/PRLLn	PPGn set High level side reload values.	PPGn set Low level side reload values.
PRLHm/PRLLm	PPGm set High level side reload values.	PPGm set Low level side reload values.

◎ : Used bit
- : Undefined bit
1 : Set 1
0 : Set 0

Note: n = C, E
m = n + 1

Note:

Use the word instruction to set both High-level and Low-level PPG reload registers (PRLLn/PRLHn and PRLLm/PRLHm) at the same time.

● Operation in 8-bit PPG output 2-channel independent operation mode

- The 8-bit PPG timer with two channels performs an independent PPG operation.
- When the pin output is enabled (PPGCn: PEC=1, PPGCm: PED=1), if the PPG output pin selection is set to standard (PPGnm:REV=0), the PPGn pulse wave is outputted from the PPGn pin and the PPGm pulse wave is outputted from the PPGm pin. When the PPG output pin is switched (PPGnm: REV=1), the PPGm pulse wave is outputted from the PPGn pin and the PPGn pulse wave is outputted from the PPGm pin.
- When the reload value is set in the PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) to enable the operation of the PPG timer (PPGCn: PENC=1, PPGCm: PEND=1), the PPG down counter of the enabled channel starts counting.
- To stop the count operation of the PPG down counter, disable the operation of the PPG timer of the channel to be stopped (PPGCn: PENC=0, PPGCm: PEND=0). The count operation of the PPG down counter is stopped and the output of the PPG output pin is held at a Low level.
- When the PPG down counter of each channel underflows, the reload values set in the PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) are reloaded to the PPG down counter that underflows.
- When an underflow occurs, the underflow generation flag bit in the channel that causes an underflow is set (PPGCn: PUFC=1, PPGCm: PUFD=1). If an interrupt request is enabled at the channel that causes an underflow (PPGCn: PIEC=1, PPGCm: PIED=1), the interrupt request is generated.

● Output waveform in 8-bit PPG output 2-channel independent operation mode

The High and Low pulse widths to be outputted are determined by adding 1 to the value in the PPG reload register and multiplying it by the count clock cycle. For example, if the value in the PPG reload register is "00_H", the pulse width has one count clock cycle, and if the value is "FF_H", the pulse width has 256 count clock cycles.

The equations for calculating the pulse width are shown below:

$\mathbf{P_L=T \times (L+1)}$

$\mathbf{P_H=T \times (H+1)}$

P_L: Low width of output pulse

P_H: High width of output pulse

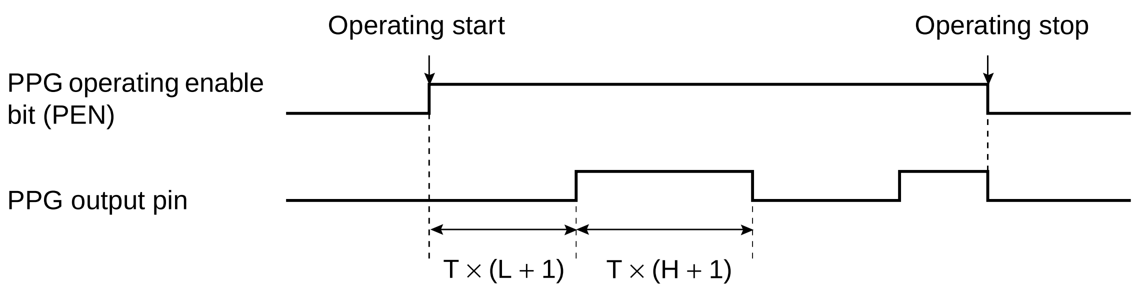
L: Values of 8 bits in PPG reload register (PRLLn or PRLLm)

H: Values of 8 bits in PPG reload register (PRLHn or PRLHm)

T: Count clock cycle

Figure 16.5-3 shows the output waveform in the 8-bit PPG output 2-channel independent operation mode.

Figure 16.5-3 Output Waveform in 8-bit PPG Output 2-channel Independent Operation Mode



L : Value of PPG reload register (PRLL)
H : Value of PPG reload register (PRLH)
T : Count clock cycle

Note: n = C, E
m = n + 1

16.5.2 16-bit PPG Output Operation Mode

In the 16-bit PPG output operation mode, the 8-/16-bit PPG timer is set as a 16-bit PPG timer with one channel.

■ Setting for 16-bit PPG Output Operation Mode

Operating the 8-/16-bit PPG timer in the 16-bit PPG output operation mode requires the setting shown in Figure 16.5-4 .

Figure 16.5-4 Setting for 16-bit PPG Output Operation Mode

	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
PPGCm/PPGCn	PEN1	–	PE1	PIE1	PUF1	MD1	MD0	Re-served	PEN0	–	PE0	PIE0	PUF0	–	–	Re-served
	1		◎	◎	◎	1	1	1	1		◎	◎	◎			1
PPGnm	(Reserved area)								PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	–	REV
									×	×	×	◎	◎	◎		×
PRLHn/PRLLn	PPGn set high level side reload values of lower 8 bits.								PPGn set low level side reload values of lower 8 bits.							
PRLHm/PRLLm	PPGm set high level side reload values of upper 8 bits.								PPGm set low level side reload values of upper 8 bits.							

◎ : Used bit
× : Unused bit
- : Undefined bit
1 : Set 1
0 : Set 0

Note: n = C, E
m = n + 1

Note:

Use a long-word instruction to set the values in the PPG reload registers or a word instruction to set the PPGn and PPGm (PRLLn --> PRLLm or PRLHn --> PRLHm) in this order.

● Operation in 16-bit PPG output operation mode

- When either PPGn pin output or PPGm pin output is enabled (PPGCn:PEC=1, PPGCm: PED=1), the same pulse wave is outputted from both the PPGn and PPGm pins.
- When the reload value is set in the PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) to enable operation of the PPG timer (PPGCn:PENC=1 and PPGCm: PEND=1), the PPG down counters start counting as 16-bit down counters (PCNTn + PCNTm).
- To stop the count operation of the PPG down counters, disable the operation of the PPG timers of both channels (PPGCn: PENC=0 and PPGCm: PEND=0). The count operation of the PPG down counters is stopped and the output of the PPG output pin is held at a Low level.
- If the PPGm down counter underflows, the reload values set in the PPGn and PPGm reload registers (PRLLn/PRLHn, PRLLm/PRLHm) are reloaded simultaneously to the PPG down counters (PCNTn + PCNTm).
- When an underflow occurs, the underflow generation flag bits in both channels are set simultaneously (PPGCn:PUFC=1, PPGCm:PUFD=1). If an interrupt request is enabled at either channel (PPGCn: PIEC=1, PPGCm: PIED=1), an interrupt request is generated.

Notes:

- In the 16-bit PPG output operation mode, the underflow generation flag bits in the two channels are set simultaneously when an underflow occurs (PPGCn: PUFC=1 and PPGCm: PUFD=1). To prevent duplication of interrupt requests, disable either of the underflow interrupt enable bits in the two channels (PPGCn:PIEC=0, PPGCm:PIED=1 or PPGCn:PIEC=1, PPGCm:PIED=0).
- If the underflow generation flag bits in the two channels are set (PPGCn: PUFC=0 and PPGCm: PUFD=0), clear the two channels at the same time.

Note: n = C, E
m = n+1

● Output waveform in 16-bit PPG output operation mode

The High and Low pulse widths to be outputted are determined by adding 1 to the value in the PPG reload register and multiplying it by the count clock cycle. For example, if the value in the PPG reload register is "0000_H", the pulse width has one count clock cycle, and if the value is "$FFFF_H$", the pulse width has 65,536 count clock cycles.

The equations for calculating the pulse width are shown below:

PL=T × (L+1)

PH=T × (H+1)

PL: Low width of output pulse

PH: High width of output pulse

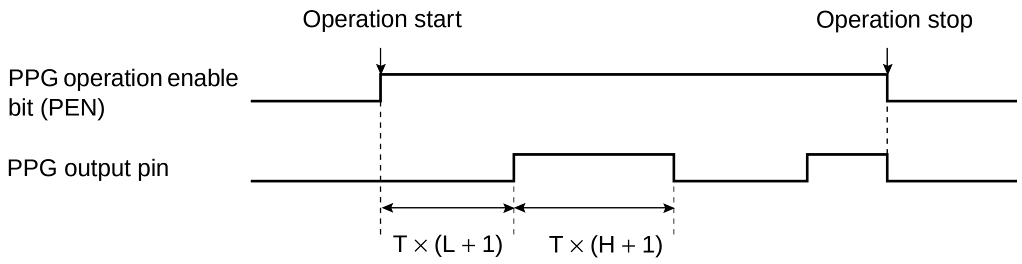
L: Values of 16 bits in PPG reload register (PRLLn+PRLLm)

H: Values of 16 bits in PPG reload register (PRLHn+PRLHm)

T: Count clock cycle

Figure 16.5-5 shows the output waveform in the 16-bit PPG output operation mode.

Figure 16.5-5 Output Waveform in 16-bit PPG Output Operation Mode



L : Values of 16 bits in PPG reload register (PRLLm + PRLLn)
H : Values of 16 bits in PPG reload register (PRLHm + PRLHn)
T : Count clock cycle

Note: n = C, E
m = n + 1

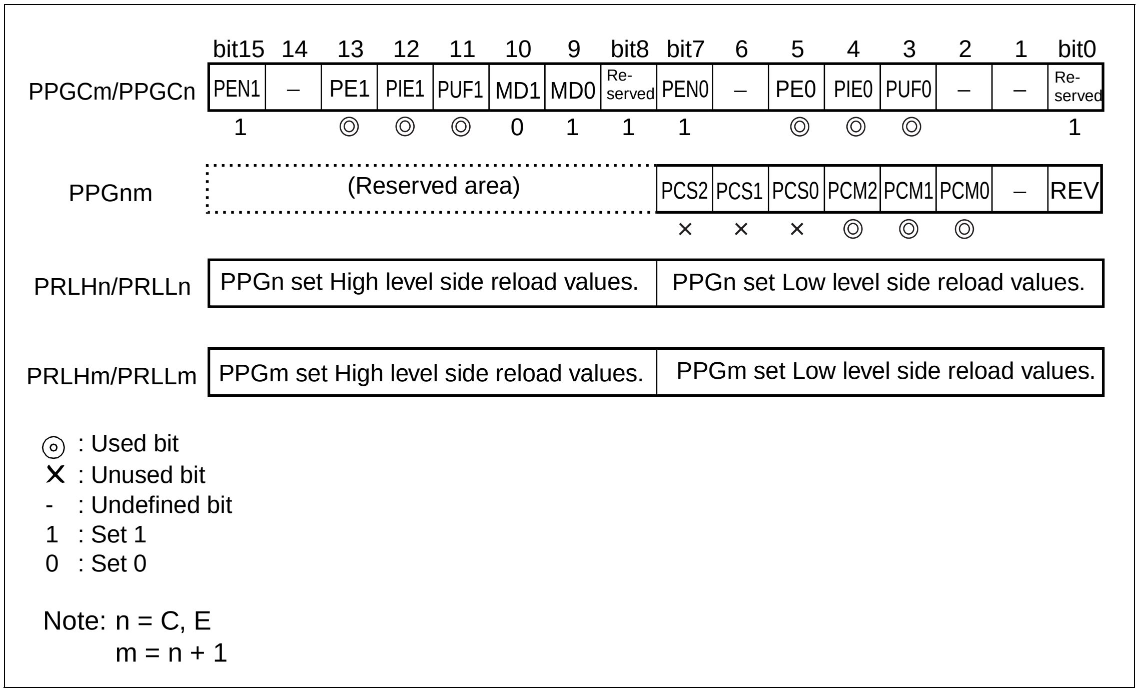
16.5.3 8+8-bit PPG Output Operation Mode

In the 8 + 8-bit PPG output operation mode, the 8-/16-bit PPG timer is set as an 8-bit PPG timer. The PPGC operates as an 8-bit prescaler and the PPG operates using the PPG output of the PPGC as a clock source.

■ Setting for 8+8-bit PPG Output Operation Mode

Operating the 8-/16-bit PPG timer in the 8+8-bit PPG output operation mode requires the setting shown in Figure 16.5-6 .

Figure 16.5-6 Setting for 8+8-bit PPG Output Operation Mode



	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
PPGCm/PPGCn	PEN1	–	PE1	PIE1	PUF1	MD1	MD0	Re-served	PEN0	–	PE0	PIE0	PUF0	–	–	Re-served
	1		◎	◎	◎	0	1	1	1		◎	◎	◎			1
PPGnm	(Reserved area)								PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	–	REV
									×	×	×	◎	◎	◎		
PRLHn/PRLLn	PPGn set High level side reload values.								PPGn set Low level side reload values.							
PRLHm/PRLLm	PPGm set High level side reload values.								PPGm set Low level side reload values.							

◎ : Used bit
× : Unused bit
- : Undefined bit
1 : Set 1
0 : Set 0

Note: n = C, E
m = n + 1

Note:

Use the word instruction to set both High-level and Low-level PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) at the same time.

● Operation in 8+8-bit PPG output operation mode

- The PPGn operates as the prescaler of the PPGm timer and the PPGm operates using the PPGn output as a clock source.
- When the pin output is enabled (PPGCn: PE0=1, PPGCm: PE1=1) if PPG output pin selection is set to standard (PPGnm:REV=0), PPGn pulse wave is outputted from the PPGn pin and the PPGm pulse wave is outputted from the PPGm pin. When the PPG output pin is switched (PPGnm:REV=1), the output pins PPGn and PPGm are switched.
- When the reload value is set in the PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) to enable operation of the PPG timer (PPGCn:PEN0=1 and PPGCm: PEN1=1), the PPG down counter starts counting.
- To stop the count operation of the PPG down counters, disable the operation of the PPG timers of both channels (PPGCn: PEN0=0 and PPGCm: PEN1=0). The count operation of the PPG down counters is stopped and the output of the PPG output pin is held at a Low level.
- If the PPG down counter of each channel underflows, the reload values set in the PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) are reloaded to the PPG down counter that underflows.
- When an underflow occurs, the underflow generation flag bit in the channel that causes an underflow (PPGCn:PUF0=1, PPGCm:PUF1=1) is set. If an interrupt request is enabled at the channel that causes an underflow (PPGCn: PIE0=1, PPGCm: PIE1=1), an interrupt request is generated.

Notes:

- Do not operate PPGm (PPGCm:PEN1 = 1) when PPGn is stopped (PPGCn:PEN0 = 0).
- It is recommended to set the same value in both Low-level and High-level PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm).

Note: n = C, E
m = n+1

- Output waveform in 8+8-bit PPG output operation mode

The High and Low pulse widths to be outputted are determined by adding 1 to the value in the PPG reload register and multiplying it by the count clock cycle.

The equations for calculating the pulse width are shown below:

PL=T × (Ln+1) × (Lm+1)

PH=T × (Hn+1) × (Hm+1)

PL: Low width of output pulse of PPGm pin

PH: High width of output pulse of PPGm pin

L_n: Values of 8 bits in PPG reload register (PRLLn)

H_n: Values of 8 bits in PPG reload register (PRLHn)

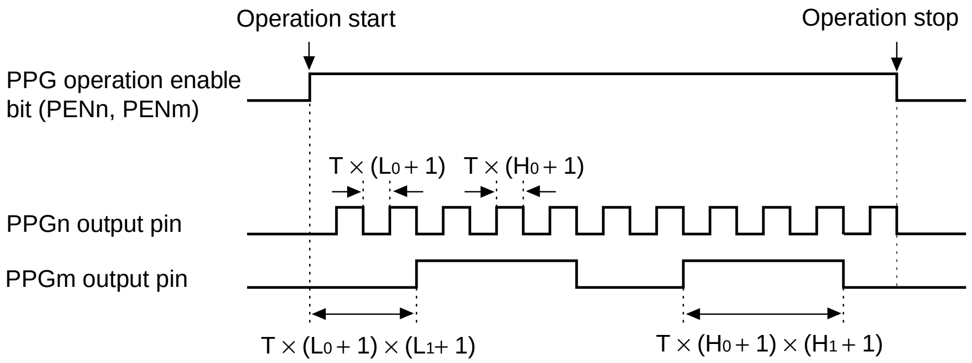
L_m: Values of 8 bits in PPG reload register (PRLLm)

H_m: Values of 8 bits in PPG reload register (PRLHm)

T: Count clock cycle

Figure 16.5-7 shows the output waveform in the 8+8-bit PPG output operation mode.

Figure 16.5-7 Output Waveform in 8+8-bit PPG Output Operation Mode



Ln : Values of 8 bits in PPG reload register (PRLLn)
Hn : Values of 8 bits in PPG reload register (PRLHn)
Hm : Values of 8 bits in PPG reload register (PRLLm)
Lm : Values of 8 bits in PPG reload register (PRLHm)
T : Count clock cycle

Note: n = C, E
m = n + 1

16.6 Precautions when Using 8-/16-bit PPG Timer

This section explains the precautions when using the 8-/16-bit PPG timer.

■ Precautions when Using 8-/16-bit PPG Timer

● Effect on 8-/16-bit PPG timer when using timebase timer output

- If the output signal of the timebase timer is used as the input signal for the count clock of the 8-/16-bit PPG timer (PPGnm: PCM2 to PCM0="111_B", PCS2 to PCS0="111_B"), deviation may occur in the first count cycle in which the PPG timer is started by trigger input or in the count cycle immediately after the PPG timer is stopped.
- When the timebase timer counter is cleared (TBTC: TBR=0) during the count operation of the PPG down counter, deviation may occur in the count cycle.

● Setting of PPG reload registers when using 8-bit PPG timer

- The Low-level and High-level pulse widths are determined at the timing of reloading the values in the Low-level PPG reload registers (PRLLn, PRLLm) to the PPG down counter.
- If the 8-bit PPG timer is used in the 8-bit PPG output 2-channel independent operation mode or the 8 + 8-bit PPG output operation mode, use a word instruction to set both High-level and Low-level PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) at the same time.

Using a byte instruction may cause an unexpected pulse to be generated.

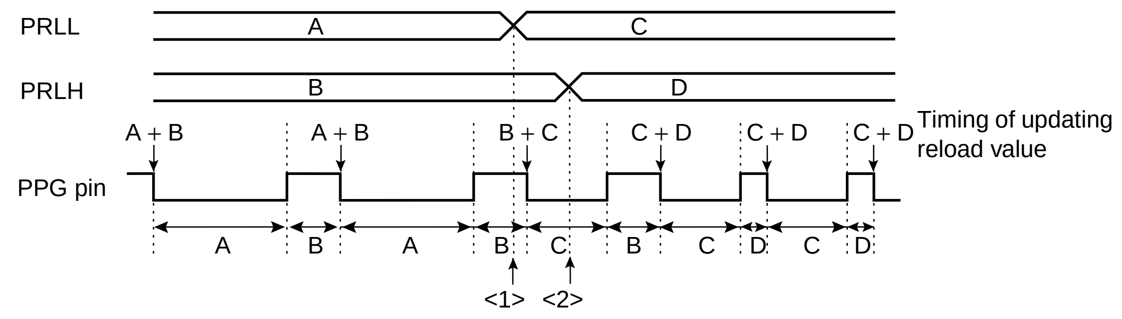
[Example of rewriting PPG reload registers using byte instruction]

Immediately before the signal level of the PPG pin switches from High to Low, if the value in the High-level PPG reload register (PRLH) is rewritten after the value in the Low-level PPG reload register (PRLL) is rewritten using the byte instruction, a Low-level pulse width is generated after rewriting and a High-level pulse width is generated before rewriting.

Figure 16.6-1 shows the waveform as the values in the PPG reload registers are rewritten using the byte instruction.

Note: n = C, E
m = n+1

Figure 16.6-1 Waveform when Values in PPG Reload Registers Rewritten Using Byte Instruction



<1>: Change the value (A → C) of PPG reload register (PRLL)
<2>: Change the value (B → D) of PPG reload register (PRLH)

● Setting of PPG reload registers when using 16-bit PPG timer

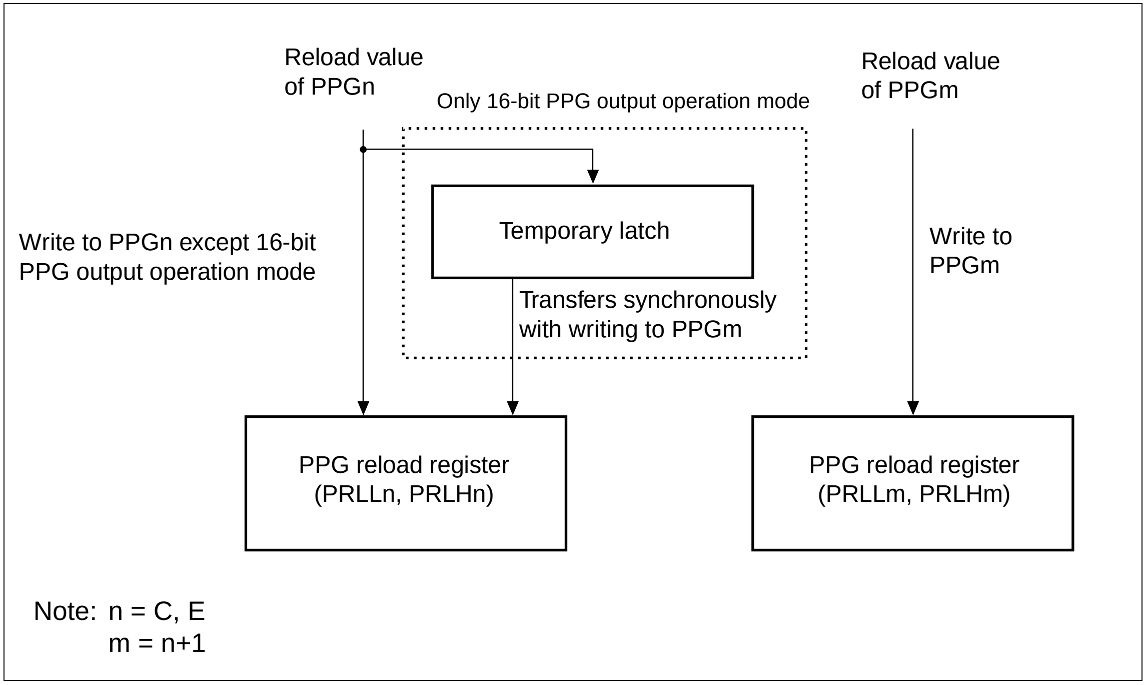
Use a long-word instruction to set the PPG reload registers (PRLLn/PRLHn, PRLLm/PRLHm) or a word instruction to set the word instruction to set the PPGn and PPGm (PRLLn --> PRLLn or PRLHm --> PRLHm) in this order.

[Reload timing in 16-bit PPG output operation mode]

In the 16-bit PPG output operation mode, the reload values written to the PPGn reload registers (PRLLn, PRLHn) are written temporarily to the temporary latch, written to the PPGm reload registers (PRLLm/ PRLHm), and then transferred to the PPGn reload registers (PRLLn/PRLHn). Therefore, when setting the reload value in the PPGm reload registers (PRLLm/PRLHm), it is necessary to set the reload value in the PPGn reload registers (PRLLn/PRLHn) simultaneously or set the reload value in the PPGn reload registers (PRLLn/PRLHn) before setting it in the PPGm reload registers (PRLLm/PRLHm).

Figure 16.6-2 shows the reload timing in the 16-bit PPG output operation mode.

Figure 16.6-2 Reload Timing in 16-bit PPG Output Operation Mode


Reload value of PPGn
Only 16-bit PPG output operation mode
Reload value of PPGm
Temporary latch
Write to PPGn except 16-bit PPG output operation mode
Write to PPGm
Transfers synchronously with writing to PPGm
PPG reload register (PRLLn, PRLHn)
PPG reload register (PRLLm, PRLHm)
Note: n = C, E
m = n+1

CHAPTER 17

DTP/EXTERNAL INTERRUPTS

This chapter explains the functions and operations of DTP/external interrupt.

17.1 Overview of DTP/External Interrupt

The DTP/external interrupt sends interrupt requests from external peripheral devices or data transfer requests to the CPU to generate an external interrupt request, or starts the EI^{2}OS.

■ DTP/External Interrupt Function

The DTP/external interrupt follows the same procedure as resource interrupts to send interrupt requests from external peripheral devices to the CPU to generate an external interrupt request, or starts the EI^{2}OS.

If the EI^{2}OS is disabled in the interrupt control register (ICR: ISE=0), the external interrupt function is enabled, branching to interrupt processing.

If the EI^{2}OS is enabled, the DTP function is enabled and automatic data transfer is performed, branching to interrupt processing after the completion of data transfer for the specified number of times.

Table 17.1-1 shows an overview of the DTP/external interrupt.

Table 17.1-1 Overview of DTP/External Interrupt

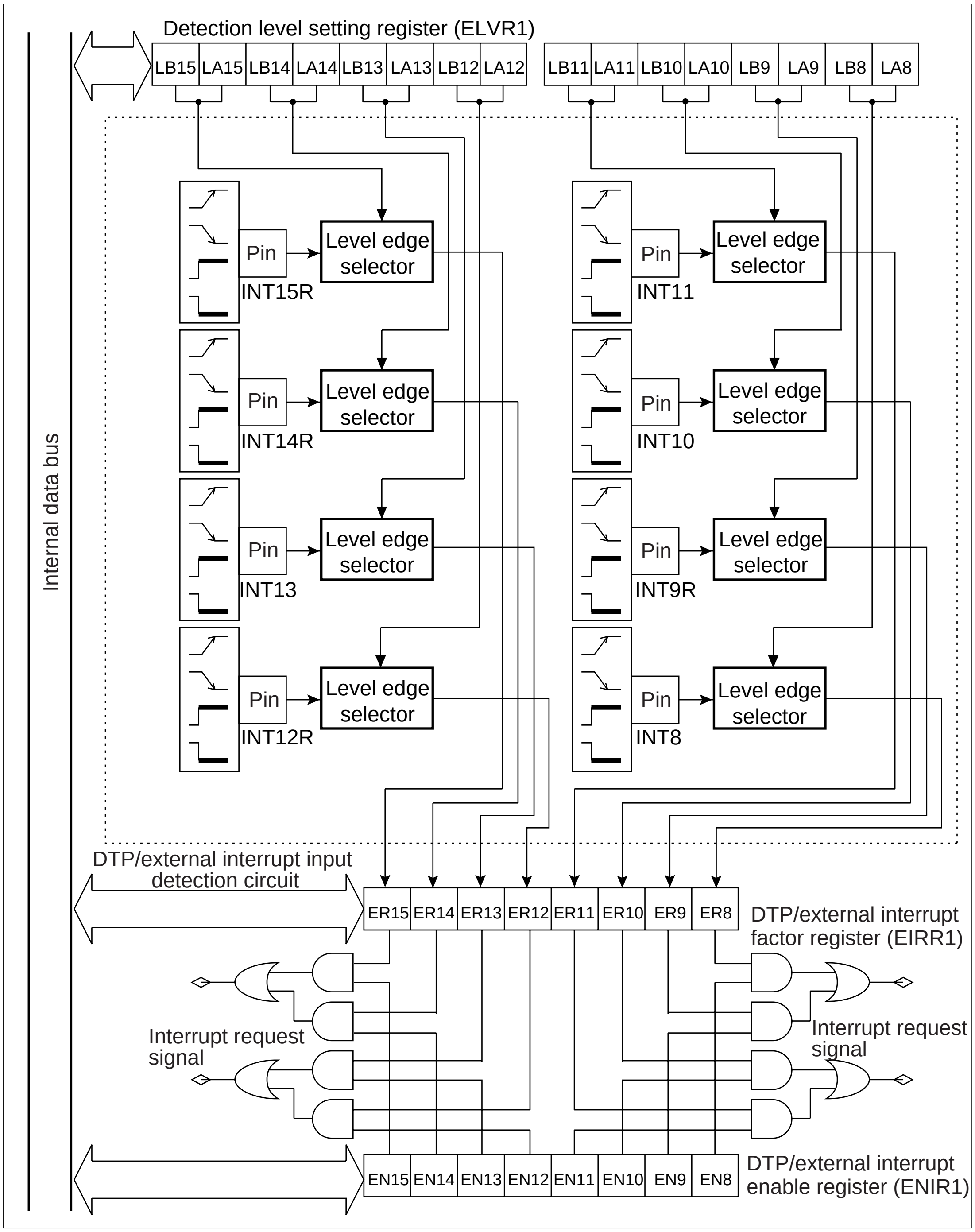
	External Interrupt	DTP Function
Input pin	8 pins : INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R, INT15R	
Interrupt factor	The interrupt factor is set in unit of pins using the detection level setting registers (ELVR1).	
	Input of High level, Low level, rising edge, or falling edge	Input of High level or Low level
Interrupt number	#26(1A$_H$), #28(1C$_H$)	
Interrupt control	The interrupt request output is enabled/disabled using the DTP/external interrupt enable register (ENIR1).	
Interrupt flag	The interrupt factor is held using the DTP/external interrupt factor register (EIRR1).	
Processing selection	The EI^{2}OS is disabled. (ICR: ISE=0)	The EI^{2}OS is enabled. (ICR: ISE=1)
Processing contents	A branch is caused to the external interrupt processing routine.	EI^{2}OS performs automatic data transfer and completes the specified number of time for data transfers, causing a branch to the interrupt processing.

17.2 Block Diagram of DTP/External Interrupt

The block diagram of the DTP/external interrupt is shown below.

■ Block Diagram of DTP/External Interrupt

Figure 17.2-1 Block Diagram of DTP/External Interrupt



- DTP/external interrupt input detection circuit

This circuit detects interrupt requests or data transfer requests generated from external peripheral devices.

The interrupt request flag bit corresponding to the pin whose level or edge set by the detection level setting register (ELVR) is detected is set to "1" (EIRR1:ER).

- Detection level setting register (ELVR1)

This register sets the level or edge of input signals from external peripheral devices that cause DTP/external interrupt factors.

- DTP/external interrupt factor register (EIRR1)

This register holds DTP/external interrupt factors.

If an enable signal is inputted to the DTP/external interrupt pin, the corresponding DTP/external interrupt request flag bit is set to "1".

- DTP/external interrupt enable register (ENIR1)

This register enables or disables DTP/external interrupt requests from external peripheral devices.

■ Details of Pins and Interrupt Numbers

Table 17.2-1 shows the pins and interrupt numbers used in the DTP/external interrupt.

Table 17.2-1 Pins and Interrupt Numbers Used by DTP/External Interrupt

Pin	Channel	Interrupt number
P54	INT8	#26(1A$_H$)
P42	INT9R	
P55	INT10	
P56	INT11	
P80	INT12R	#28(1C$_H$)
P57	INT13	
P82	INT14R	
P84	INT15R	

INT9R, INT12R, INT14R, and INT15R are enabled by setting the corresponding bit of the external interrupt factor select register (EISSR) to "1".

17.3 Configuration of DTP/External Interrupt

This section lists and details the pins, interrupt factors, and registers in the DTP/external interrupt.

■ Pins of DTP/External Interrupt

The pins used by the DTP/external interrupt serve as general-purpose I/O ports.

Table 17.3-1 lists the pin functions and the pin setting required for use in the DTP/external interrupt.

Table 17.3-1 Pins of DTP/External Interrupt

Pin Name	Pin Function	Pin Settings Required for Use in DTP/ External Interrupt
P54/AN12/TOT3/INT8	General-purpose I/O ports, analog input, event pin for reload timer, DTP external interrupt inputs	• Set external interrupt factor select register (EISSR) to 0. • Set as input ports in port direction register (DDR5).
P55/AN13/INT10	General-purpose I/O ports, analog input, DTP external interrupt inputs	
P56/AN14/INT11		
P57/AN15/INT13		
P42/INT9R/RX1/	General-purpose I/O ports, DTP external interrupt inputs, CAN1 input Rx1	• Set external interrupt factor select register (EISSR) to 1. • Set as input ports in port direction register (DDR4).
P80/INT12R/ADTG	General-purpose I/O ports, DTP external interrupt inputs, A/D converter trigger input ADTG	• Set external interrupt factor select register (EISSR) to 1. • Set as input ports in port direction register (DDR8).
P82/INT14R/SIN0/TIN2	General-purpose I/O ports, DTP external interrupt inputs, UART0 input SIN0, reload timer 2 trigger input TIN2	
P84/INT15R/SCK0	General-purpose I/O ports, DTP external interrupt inputs, UART0 clock I/O SCK0	

■ List of Registers and Reset Values in DTP/External Interrupt

Figure 17.3-1 List of Registers and Reset Values in DTP/External Interrupt

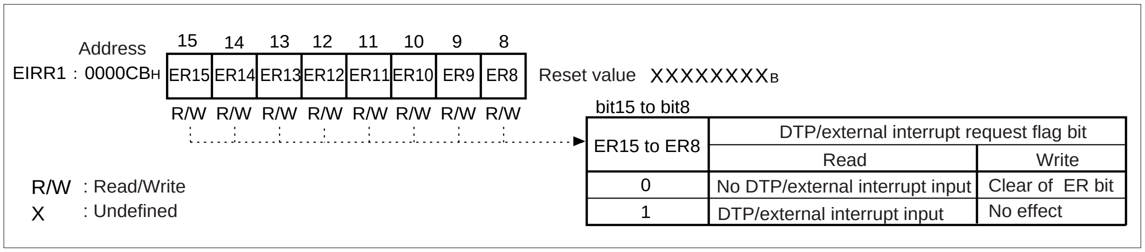
										Reset value
ENIR1	bit	7	6	5	4	3	2	1	0	
Address: 0000CA$_H$		EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	00000000$_B$
EIRR1	bit	15	14	13	12	11	10	9	8	
Address: 0000CB$_H$		ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8	XXXXXXXX$_B$
ELVR1	bit	7	6	5	4	3	2	1	0	
Address: 0000CC$_H$		LB11	LA11	LB10	LA10	LB9	LA9	LB8	LA8	00000000$_B$
ELVR1	bit	15	14	13	12	11	10	9	8	
Address: 0000CD$_H$		LB15	LA15	LB14	LA14	LB13	LA13	LB12	LA12	00000000$_B$
EISSR	bit	7	6	5	4	3	2	1	0	
Address: 0000CE$_H$		INT15R	INT14R	INT13R	INT12R	INT11R	INT10R	INT9R	INT8R	00000000$_B$

17.3.1 DTP/External Interrupt Factor Register (EIRR1)

The DTP/external interrupt factor register holds DTP/external interrupt factors.
When a valid signal is inputted to the DTP/external interrupt pin, the corresponding DTP/external interrupt request flag bit is set to "1".
The EIRR1 register is corresponding to INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R, and INT15R.

■ DTP/External Interrupt Factor Register (EIRR1)

Figure 17.3-2 DTP/External Interrupt Factor Register (EIRR1)



ER15 to ER8	DTP/external interrupt request flag bit	
	Read	Write
0	No DTP/external interrupt input	Clear of ER bit
1	DTP/external interrupt input	No effect

Table 17.3-2 Function of DTP/External Interrupt Factor Register (EIRR1)

Bit Name		Function
bit8 to bit15	ER15 to ER8(EIRR1), DTP/External interrupt request flag bits	These bits are set to "1" when the edges or level signals set by the detection condition select bits in the detection level setting register (ELVR1:LB, LA) are inputted to the DTP/external interrupt pins. **When set to "1":** When the DTP/external interrupt request enable bit (ENIR1:EN) is set to "1", an interrupt request is generated to the corresponding DTP/external interrupt channel. **When set to "0":** Cleared **When set to "1":** No effect **Note:** Reading by read-modify-write type instructions always returns "1". If more than one DTP/external interrupt request is enabled (ENIR1:EN = 1), clear only the bit in the channel that accepts an interrupt (EIRR1:ER = 0). No other bits must be cleared unconditionally. **Reference:** When the EI^{2}OS is started, the interrupt request flag bit is automatically cleared after the completion of data transfer (EIRR1:ER = 0).

17.3.2 DTP/External Interrupt Enable Register (ENIR1)

The DTP/external interrupt enable register (ENIR1) enables/disables the DTP/external interrupt request in the external peripheral devices.
ENIR1 is corresponding to INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R and INT15R.

■ DTP/External Interrupt Enable Register (ENIR1)

Figure 17.3-3 DTP/External Interrupt Enable Register (ENIR1)

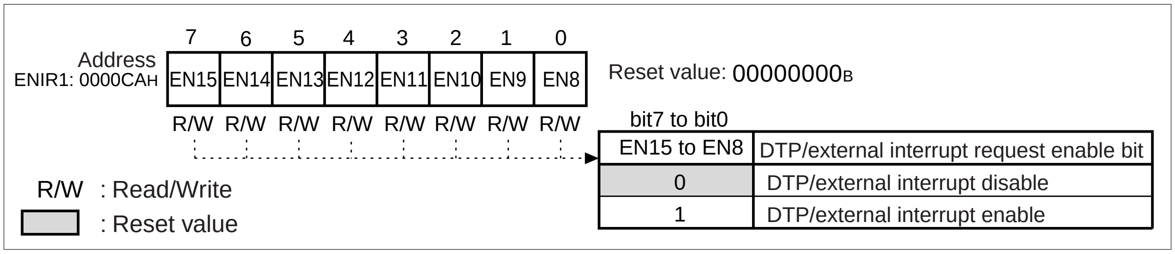

7 6 5 4 3 2 1 0
Address
ENIR1: 0000CAH
EN15 EN14 EN13 EN12 EN11 EN10 EN9 EN8
Reset value: 00000000B
R/W R/W R/W R/W R/W R/W R/W R/W
bit7 to bit0
EN15 to EN8
DTP/external interrupt request enable bit
0
DTP/external interrupt disable
1
DTP/external interrupt enable
R/W : Read/Write
: Reset value

Table 17.3-3 Functions of DTP/External Interrupt Enable Register (ENIR1)

Bit Name		Function
bit0 to bit7	EN15 to EN8(ENIR1), DTP/external interrupt request enable bits	These bits enable or disable the DTP/external interrupt request to the DTP/external interrupt channel. If the DTP/external interrupt request enable bit (ENIR1:EN) and the DTP/external interrupt request flag bit (EIRR1:ER) are set to "1", the interrupt request is generated to the corresponding DTP/external interrupt pin. **Reference:** The state of the DTP/external interrupt pin can be read directly using the port data register irrespective of the setting of the DTP/external interrupt request enable bit.

Table 17.3-4 Correspondence between DTP/External Interrupt Pins, DTP/External Interrupt Request Flag Bits, and DTP/External Interrupt Request Enable Bits

DTP/external interrupt pin	DTP/external interrupt request flag bit	DTP/external interrupt request enable bit
INT8	ER8	EN8
INT9R	ER9	EN9
INT10	ER10	EN10
INT11	ER11	EN11
INT12R	ER12	EN12
INT13	ER13	EN13
INT14R	ER14	EN14
INT15R	ER15	EN15

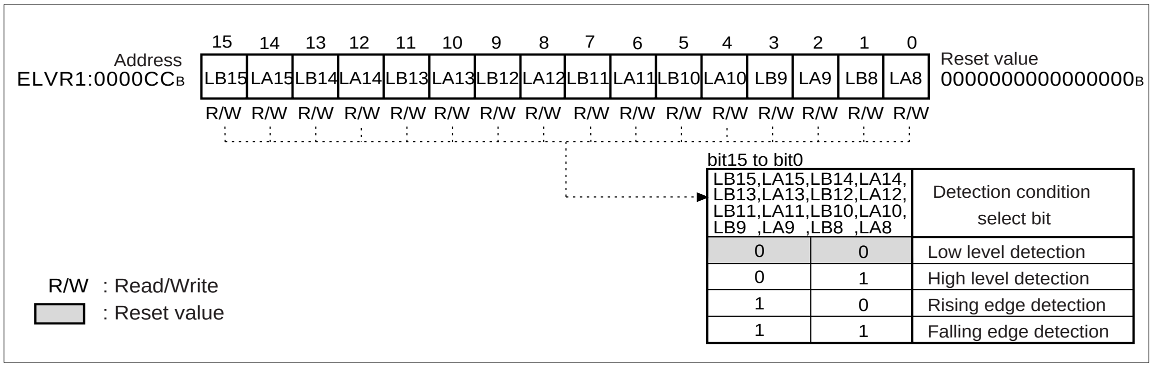
17.3.3 Detection Level Setting Register (ELVR1)

The detection level setting register sets the level or edge of input signals that cause the interrupt factors of the DTP/external interrupt pin.

ELVR1 is corresponding to INT8, INT9R, INT10, INT11, INT12R, INT13, INT14R and INT15R.

■ Detection Level Setting Register (ELVR1)

Figure 17.3-4 Detection Level Setting Register (ELVR1)


Address
ELVR1:0000CCB
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
LB15 LA15 LB14 LA14 LB13 LA13 LB12 LA12 LB11 LA11 LB10 LA10 LB9 LA9 LB8 LA8
R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W
Reset value
0000000000000000B
bit15 to bit0
R/W : Read/Write
: Reset value

LB15,LA15,LB14,LA14, LB13,LA13,LB12,LA12, LB11,LA11,LB10,LA10, LB9 ,LA9 ,LB8 ,LA8		Detection condition select bit
0	0	Low level detection
0	1	High level detection
1	0	Rising edge detection
1	1	Falling edge detection

Table 17.3-5 Functions of Detection Level Setting Register (ELVR1)

Bit Name		Function
bit15 to bit0	ELVR1 ... LB15, LA15 to LB8, LA8 Detection condition select bits	These bits set the levels or edges of input signals from external peripheral devices that cause interrupt factors in the DTP/external interrupt pins. • Two levels or two edges are selectable for external interrupts, and two levels are selectable for the EI^{2}OS. **Reference:** When the set detection signal is input to the DTP/external interrupt pins, the DTP/external interrupt request flag bits are set to "1" even if DTP/external interrupt requests are disabled (ENIR1:EN = 0).

Table 17.3-6 Correspondence between Detection Level Setting Register and Channels

DTP/External Interrupt Pin	Register Name	Bit Name
INT8	ELVR1	LB8, LA8
INT9R		LB9, LA9
INT10		LB10, LA10
INT11		LB11, LA11
INT12R		LB12, LA12
INT13		LB13, LA13
INT14R		LB14, LA14
INT15R		LB15, LA15

17.3.4 External Interrupt Factor Select Register (EISSR)

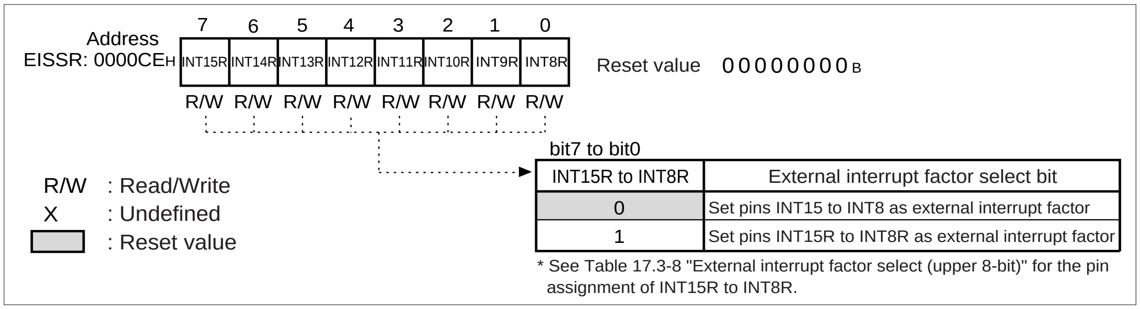
The external interrupt factor select register (EISSR) can change the assignment of the external interrupt pin. This allows the external interrupt. Also, the function such as CAN wakeup is implemented.

■ Selection of External Interrupt Factor

The external interrupt pin of the upper 8-bit is assigned to INT13, INT11, INT10, and INT8 normally and shares the port 5 and pin. In the external bus mode, the port 0 cannot be used as the external interrupt pin. The pin is switched by the external interrupt factor select register (EISSR). In addition, because INT15R, INT14R, INT12R, and INT9R share the function such as CAN input pin, the function such as CAN wakeup can be implemented.

See Table 17.3-8 for the pin function of INT15R, INT14R, INT12R, and INT9R.

Figure 17.3-5 DTP/external Interrupt Factor Select Register (EISSR)


Address
EISSR: 0000CEH
7 6 5 4 3 2 1 0
INT15R INT14R INT13R INT12R INT11R INT10R INT9R INT8R
R/W R/W R/W R/W R/W R/W R/W R/W
Reset value 00000000B
bit7 to bit0
INT15R to INT8R
External interrupt factor select bit
0
Set pins INT15 to INT8 as external interrupt factor
1
Set pins INT15R to INT8R as external interrupt factor
R/W : Read/Write
X : Undefined
: Reset value

* See Table 17.3-8 "External interrupt factor select (upper 8-bit)" for the pin assignment of INT15R to INT8R.

Table 17.3-7 Function of DTP/external Interrupt Factor Select Register (EISSR)

Bit Name		Function
bit7 to bit0	INT15R to INT8R: External interrupt factor select bits	When these bits are set to "1", the input pin of the corresponding external interrupt factor (upper 8-bit) is assigned to the INT15R to INT8R. **When set to "0" :** The external interrupt factor of the upper 8-bit is assigned to INT15 to INT8 pins. **When set to "1":** The external interrupt factor of the upper 8-bit is assigned to the INT15R to INT8R pins.

Table 17.3-8 External Interrupt Factor Select (Upper 8-bit)

EISSR Bit	"0" (Initial Value)	"1"
INT8R	INT8: P54 (AN12/TOT3)	-
INT9R	-	INT9R: P42 (RX1)
INT10R	INT10: P55 (AN13)	-
INT11R	INT11: P56 (AN14)	-
INT12R	-	INT12R: P80 (ADTG)
INT13R	INT13: P57 (AN15)	-
INT14R	-	INT14R: P82 (SIN0/TIN2)
INT15R	-	INT15R: P84 (SCK0)

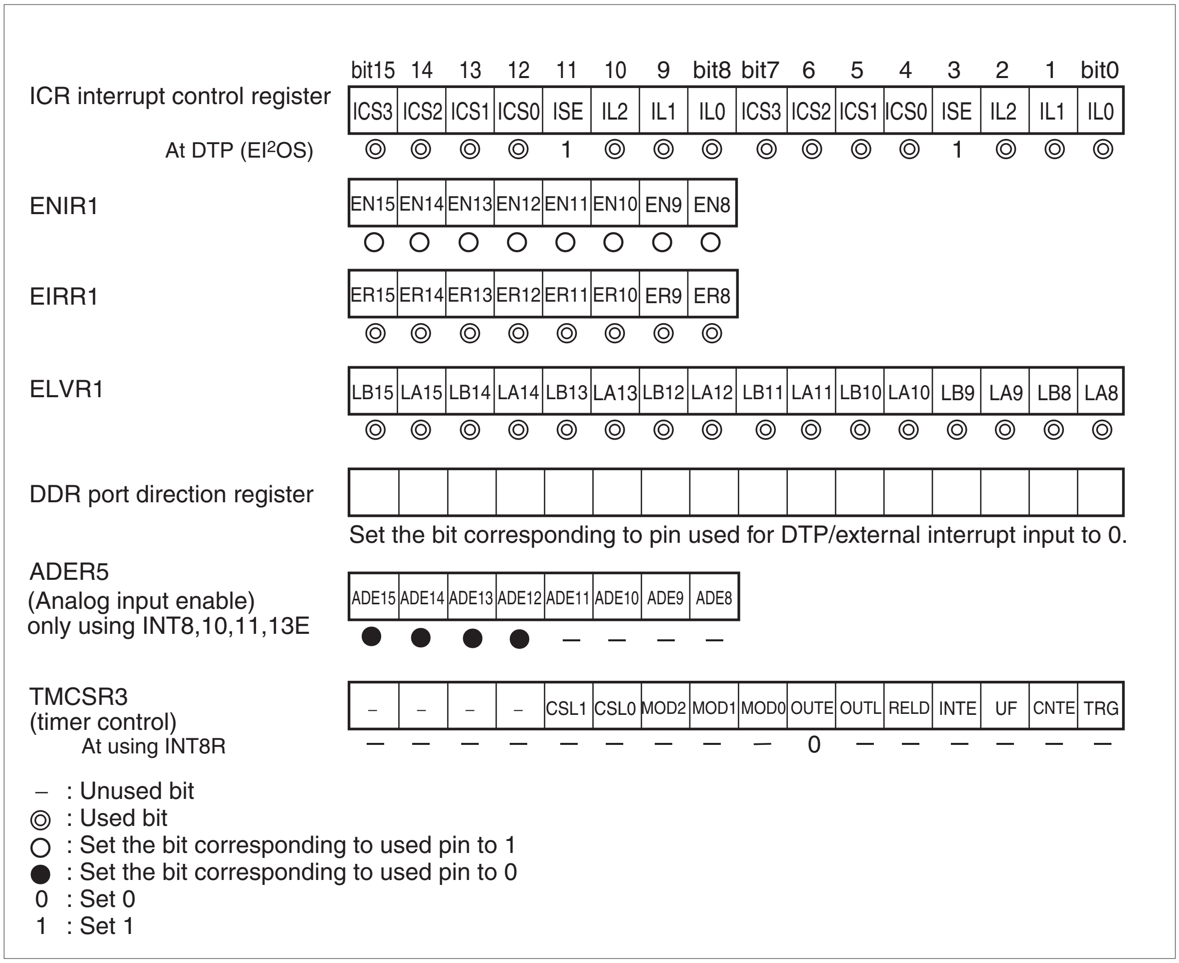
17.4 Explanation of Operation of DTP/External Interrupt

The DTP/external interrupt has an external interrupt function and a DTP function. The setting and operation of each function are explained.

■ Setting of DTP/External Interrupt

Using the DTP/external interrupt requires, the setting shown in Figure 17.4-1 .

Figure 17.4-1 Setting of DTP/External Interrupt



	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
ICR interrupt control register	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0
At DTP (EI²OS)	◎	◎	◎	◎	1	◎	◎	◎	◎	◎	◎	◎	1	◎	◎	◎
ENIR1	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8								
	○	○	○	○	○	○	○	○								
EIRR1	ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8								
	◎	◎	◎	◎	◎	◎	◎	◎								
ELVR1	LB15	LA15	LB14	LA14	LB13	LA13	LB12	LA12	LB11	LA11	LB10	LA10	LB9	LA9	LB8	LA8
	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎	◎
DDR port direction register																
ADER5 (Analog input enable) only using INT8,10,11,13E	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8								
	●	●	●	●	–	–	–	–								
TMCSR3 (timer control)	–	–	–	–	CSL1	CSL0	MOD2	MOD1	MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
At using INT8R	–	–	–	–	–	–	–	–	–	0	–	–	–	–	–	–

Set the bit corresponding to pin used for DTP/external interrupt input to 0.

– : Unused bit
◎ : Used bit
○ : Set the bit corresponding to used pin to 1
● : Set the bit corresponding to used pin to 0
0 : Set 0
1 : Set 1

● Setting procedure

To use the DTP/external interrupt, set each register by using the following procedure:

1. Set the input port to the general-purpose I/O port, which is shared with the terminal to be used as external interrupt input.
2. Set the external interrupt factor select register (EISSR) corresponding to the DTP/external interrupt channel to be used.
3. Set the interrupt request enable bit corresponding to the DTP/external interrupt channel to be used to 0 (ENIR1:EN).
4. Use the detection condition select bit corresponding to the DTP/external interrupt pin to be used to set the edge or level to be detected (ELVR1: LA, LB).
5. Set the interrupt request flag bit corresponding to the DTP/external interrupt channel to be used to 0 (EIRR1: ER).
6. Set the interrupt request enable bit corresponding to the DTP/external interrupt channel to be used to 1 (ENIR1: EN).

Note that concurrent writing with 16-bit data is available in 5 and 6.

- When setting the registers for the DTP/external interrupt, the external interrupt request must be disabled in advance (ENIR1: EN = 0).
- When enabling the DTP/external interrupt request (ENIR1:EN = 1), the corresponding DTP/external interrupt request flag bit must be cleared in advance (EIRR1:ER = 0). These actions prevent the mistaken interrupt request from occurring when setting the register.

● Selecting of DTP or external interrupt function

Whether the DTP function or the external interrupt function is executed depends on the setting of the EI^2OS enable bit in the corresponding interrupt control register (ICR:ISE).

If the ISE bit is set to "1", the EI^2OS is enabled.

If the ISE bit is set to "0", the EI^2OS is disabled and the external interrupt function is executed.

Notes:

- All interrupt requests assigned to one interrupt control register have the same interrupt levels (IL2 to IL0).
- If two or more interrupt requests are assigned to one interrupt control register and the EI^2OS is used in one of them, other interrupt requests cannot be used.
- Enabling unequipped terminals causes a false operation. First set the EISSR and then set each of the registers when DTP/external interrupt is used.

■ DTP/External Interrupt Operation

The control bits and the interrupt factors for the DTP/external interrupt are shown in Table 17.4-1 .

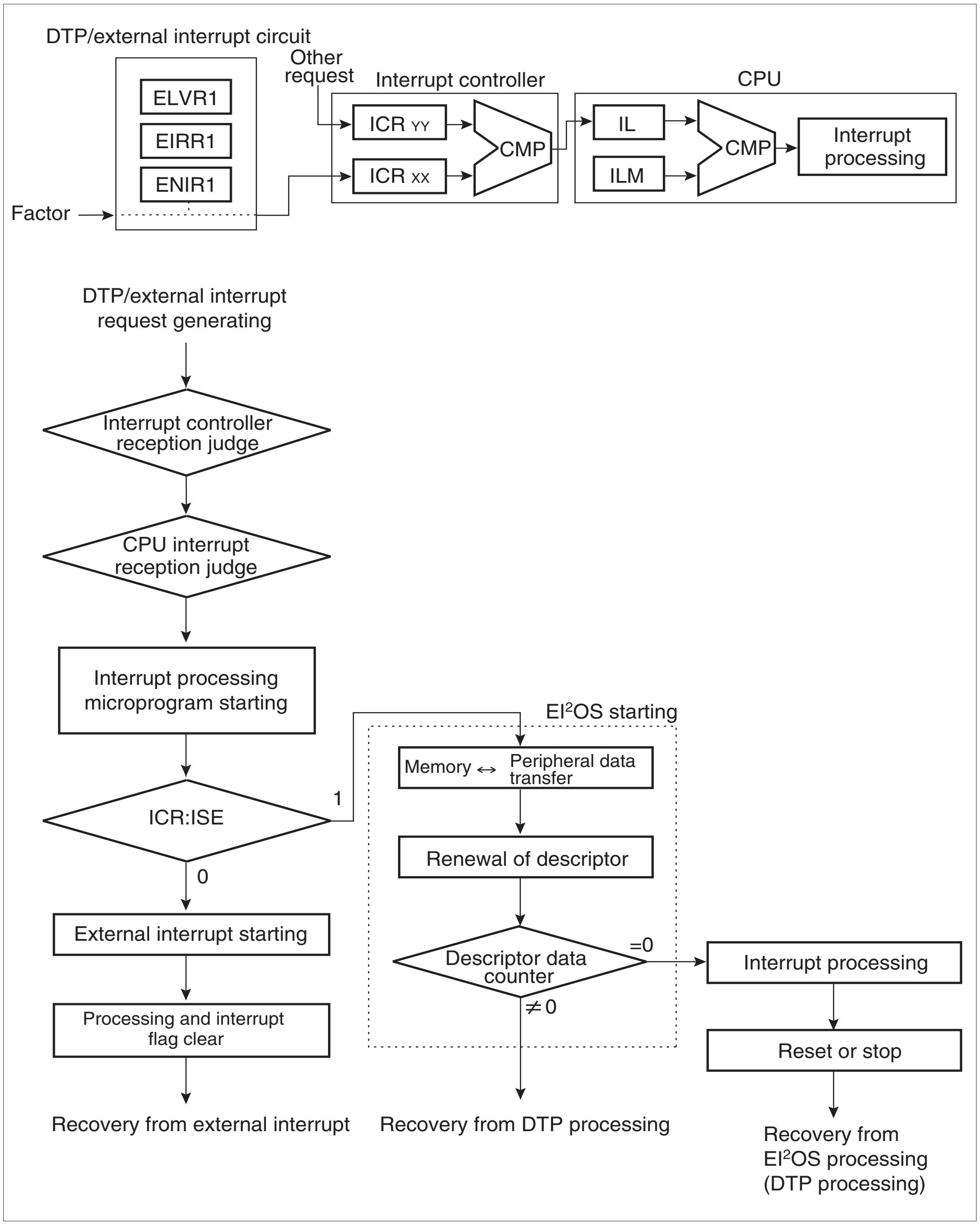
Table 17.4-1 Control Bits and Interrupt Factors for DTP/External Interrupt

	DTP/External interrupt
Interrupt request flag bit	EIRR1: ER15 to ER8
Interrupt request enable bit	ENIR1: EN15 to EN8
Interrupt factor	Input of valid edge or level to INT13, INT11, INT10, INT8 , INT9R, INT12R, INT14R, INT15R pins

If the interrupt request signal from the DTP/external interrupt is output to the interrupt controller and the EI^{2}OS enable bit in the interrupt control register (ICR:ISE) is set to "0", the interrupt processing is executed. This bit is set to "1", the EI^{2}OS is executed.

Figure 17.4-2 shows the operation of the DTP/external interrupt.

Figure 17.4-2 Operation of DTP/External Interrupt


DTP/external interrupt circuit
ELVR1
EIRR1
ENIR1
Factor
Other request
Interrupt controller
ICR YY
ICR XX
CMP
CPU
IL
ILM
CMP
Interrupt processing
DTP/external interrupt request generating
Interrupt controller reception judge
CPU interrupt reception judge
Interrupt processing microprogram starting
ICR:ISE
1
0
External interrupt starting
Processing and interrupt flag clear
Recovery from external interrupt
EI²OS starting
Memory ↔ Peripheral data transfer
Renewal of descriptor
Descriptor data counter
=0
≠0
Recovery from DTP processing
Interrupt processing
Reset or stop
Recovery from EI²OS processing (DTP processing)

17.4.1 External Interrupt Function

The DTP/external interrupt has an external interrupt function for generating an interrupt request by detecting the signal (edge or level) in the DTP/external interrupt pin.

■ External Interrupt Function

- When the signal (edge or level) set in the detection level setting register is detected in the DTP/external interrupt pin, the interrupt request flag bit in the DTP/external interrupt factor register (EIRR1:ER) is set to "1".
- If the interrupt request enable bit in the DTP/external interrupt enable register is enabled (ENIR1:EN = 1) with the interrupt request flag bit set to "1", the interrupt request generation is posted to the interrupt controller.
- If an interrupt request is preferred to other interrupt request by the interrupt controller, the interrupt request is generated.
- If the level of an interrupt request (ICR:IL) is higher than that of the interrupt level mask bit in the condition code register (CCR:ILM) and the interrupt enable bit is enabled (PS:CCR:I = 1), the CPU performs interrupt processing after completion of the current instruction execution and branches to interrupt processing.
- At interrupt processing, set the corresponding DTP/external interrupt request flag bit to 0 and clear the DTP/external interrupt request.

Notes:

- When the DTP/external interrupt start factor is generated, the DTP/external interrupt request flag bit (EIRR1:ER) is set to "1", regardless of the setting of the DTP/external interrupt request enable bit (ENIR1:EN).
- When the interrupt processing is started, clear the DTP/external interrupt request flag bit that caused the start factor. Control cannot be returned from the interrupt while the DTP/external interrupt request flag bit is set to "1". When clearing, do not clear any flag bit other than the accepted DTP/external interrupt factor.

17.4.2 DTP Function

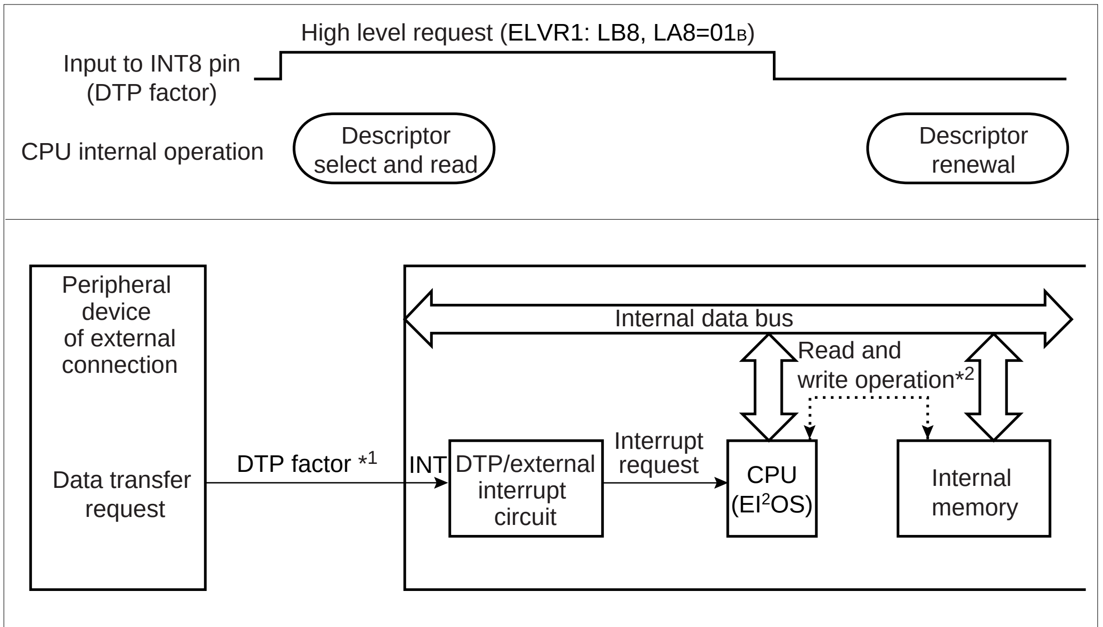
The DTP/external interrupt has the DTP function that detects the signal of the external peripheral device from the DTP/external interrupt pin to start the EI^{2}OS.

■ DTP Function

The DTP function detects the signal level set by the detection level setting register of the DTP/external interrupt function to start the EI^{2}OS.

- When the EI^{2}OS operation is already enabled (ICR:ISE = 1) at the point when the interrupt request is accepted by the CPU, the DTP function starts the EI^{2}OS and starts data transfer.
- When transfer of one data item is completed, the descriptor is updated and the DTP/external interrupt request flag bit is cleared to prepare for the next request from the DTP/external interrupt pin.
- When the EI^{2}OS completes transfer of all the data, control branches to the interrupt processing.

Figure 17.4-3 Example of Interface with External Peripheral Device (when using EI^{2}OS in Single-chip mode)


High level request (ELVR1: LB8, LA8=01B)
Input to INT8 pin (DTP factor)
CPU internal operation
Descriptor select and read
Descriptor renewal
Peripheral device of external connection
Data transfer request
DTP factor *1
INT
DTP/external interrupt circuit
Interrupt request
CPU (EI2OS)
Internal memory
Internal data bus
Read and write operation*2

*1: This must be cancelled within three machine clocks after the start of data transfer.
*2: When EI^{2}OS is "peripheral function" → "internal memory transfer".

17.5 Precautions when Using DTP/External Interrupt

This section explains the precautions when using the DTP/external interrupt.

■ Precautions when Using DTP/External Interrupt

● Condition of external-connected peripheral device when DTP function is used

- When using the DTP function, the peripheral device must automatically clear a data transfer request when data transfer is performed.
- Inactivate the transfer request signal within three machine clocks after starting data transfer. If the transfer request signal remains active, the DTP/external interrupt regards the transfer request signal as a generation of next transfer request.

● External interrupt input polarity

- When the edge detection is set in the detection level setting register, the pulse width for edge detection must be at least three machine clocks.
- When a level causing an interrupt factor is inputted with level detection set in the detection level setting register, factor F/F in the DTP/external interrupt factor register is set to "1" and the factor is held as shown in Figure 17.5-1 .

With the factor held in factor F/F, the request to the interrupt controller remains active if the interrupt request is enabled (ENIR1: EN = 1) even after the DTP/external interrupt factor is cancelled. To cancel the request to the interrupt controller, clear the external interrupt request flag bit (EIRR1: ER) and clear the factor F/F as shown in Figure 17.5-2 .

Figure 17.5-1 Clearing Factor Hold Circuit when Level Set

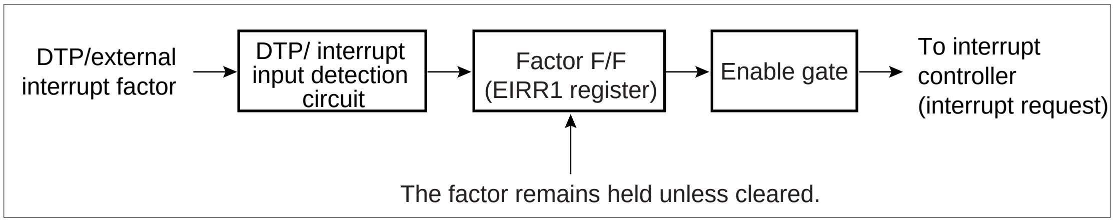
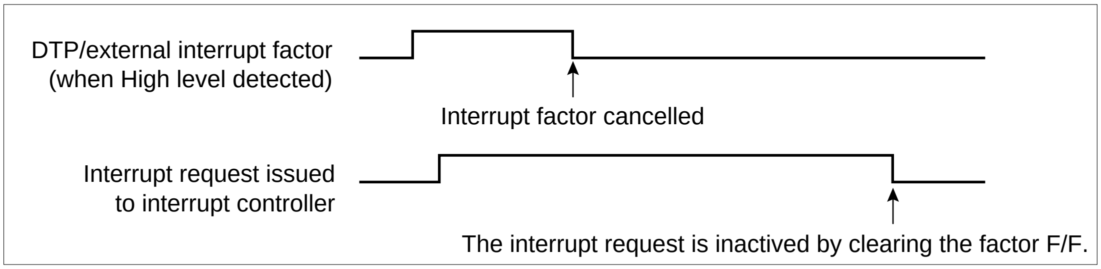

DTP/external interrupt factor
DTP/ interrupt input detection circuit
Factor F/F (EIRR1 register)
Enable gate
To interrupt controller (interrupt request)
The factor remains held unless cleared.

Figure 17.5-2 DTP/External Interrupt Factor and Interrupt Request Generated when Interrupt Request Enabled


DTP/external interrupt factor (when High level detected)
Interrupt factor cancelled
Interrupt request issued to interrupt controller
The interrupt request is inactived by clearing the factor F/F.

● Precautions on interrupts

- When the DTP/external interrupt is used as the external interrupt function, no return from interrupt processing can be made with the DTP/external interrupt request flag bit set to "1" (EIRR1:ER) and the DTP/external interrupt request set to "enabled" (ENIR1:EN = 1). Always set the DTP/external interrupt request flag bit to 0 (EIRR1:ER) at interrupt processing.
- When the level detection is set in the detection level setting register and the level that becomes the interrupt factor remains input, the DTP/external interrupt request flag bit is reset immediately even when cleared (EIRR1:ER = 0). Disable the DTP/external interrupt request output as needed (ENIR1:EN = 0), or cancel the interrupt factor itself.

17.6 Program Example of DTP/External Interrupt Function

This section gives a program example of the DTP/external interrupt function.

■ Program Example of DTP/External Interrupt Function

● Processing specifications

An external interrupt is generated by detecting the rising edge of the pulse input to the INT8 pin.

● Coding example

```
ICR07  EQU   0000B7H            ;Interrupt control register ICR7
DDR5   EQU   000015H            ;Port 5 direction register
ENIR1  EQU   0000CAH            ;DTP/external interrupt enable
                                 register 1
EIRR1  EQU   0000CBH            ;DTP/external interrupt factor
                                 register 1
ELVR1L EQU   0000CCH            ;Detection level setting register 1:"L"
ELVR1H EQU   0000CDH            ;Detection level setting register 1:"H"
ADER5  EQU   00000BH            ;Port5 analog input enable register
ER8    EQU   EIRR1:0            ;INT8 Interrupt request flag bit
EN8    EQU   ENIR1:0            ;INT8 Interrupt request enable bit
;---------Main program------------------------------------
CODE   CSEG
START:                 ;Stack pointer (SP) already initialized
       MOV   I:ADER5,#00000000B ;Set analog input of Port5 to disable
       MOV   I:DDR5,#00000000B  ;Set DDR5 to input port
       AND   CCR,#0BFH          ;Interrupts disabled
       MOV   I:ICR07,#00H       ;Interrupt level 0 (highest)
       CLRB  I:EN8              ;INT8 disabled using ENIR1
       MOV   I:ELVR0L,#00000010B;Rising edge selected for INT8
       CLRB  I:ER0              ;INT8 interrupt flag cleared using
                                ;EIRR1
       SETB  I:EN8              ;INT8 interrupt request enabled using
                                 ENIR1
       MOV   ILM,#07H           ;Set ILM in PS to level 7
       OR    CCR,#40H           ;Interrupts enabled
LOOP:
       ÅE
       Processing by user
       ÅE
       BRA   LOOP
;---------Interrupt program-------------------------------------
WARI:
       CLRB   I:ER8              ;Interrupt request flag cleared
       ÅE
        Processing by user
       ÅE
       RETI                    ;Return from interrupt processing
CODE   ENDS
;---------Vector setting---------------------------------------
VECT   CSEG  ABS=0FFH
       ORG   00FF94H           ;Set vector to interrupt number
                                #26(1AH)
       DSL   WARI
       ORG   00FFDCH           ;Reset vector set
       DSL   START
       DB    00H               ;Set to single-chip mode
VECT   ENDS
       END   START
```

■ Program Example of DTP Function

● Processing specification

- Channel 0 of the EI^{2}OS is started by detecting the High level of the signal input to the INT8 pin.
- RAM data is outputted to port 5 by performing DTP processing (EI^{2}OS).

● Coding example

```
ICR07  EQU   0000B7H           ;DTP/external interrupt control
                                register
DDR6   EQU   000016H           ;Port 6 direction register
DDR5   EQU   000015H           ;Port 5 direction register
ENIR1  EQU   0000CAH           ;DTP/external interrupt enable
                                register 1
EIRR1  EQU   0000CBH           ;DTP/external interrupt factor
                                register 1
ELVR1L EQU   0000CCH           ;Detection level setting register 1:"L"
ELVR1H EQU   0000CDH           ;Detection level setting register 1:"H"
ADER5  EQU   00000BH           ;Port5 analog input enable register
ADER6  EQU   00000CH           ;Port6 analog input enable register
ER1    EQU   EIRR:0            ;INT8 interrupt request flag bit
EN1    EQU   ENIR:0            ;INT8 interrupt request enable bit
;
BAPL   EQU   000100H           ;Buffer address pointer lower
BAPM   EQU   000101H           ;Buffer address pointer middle
BAPH   EQU   000102H           ;Buffer address pointer higher
ISCS   EQU   000103H           ;EI2OS status register
IOAL   EQU   000104H           ;I/O address register lower
IOAH   EQU   000105H           ;I/O address register higher
DCTL   EQU   000106H           ;Data counter lower
DCTH   EQU   000107H           ;Data counter higher
```

```
;
;---------Main program------------------------------------
CODE   CSEG
START:                ;Stack pointer (SP) already initialized
       MOV   I:ADER5,#00000000B ;Set analog input of port5 to disable
       MOV   I:ADER6,#00000000B ;Set analog input of port6 to disable
       MOV   I:DDR6,#11111111B  ;Set DDR6 to output port
       MOV   I:DDR5,#00000000B  ;Set DDR5 to input port
       AND   CCR,#0BFH          ;Interrupts disabled
       MOV   I:ICR07,#08H       ;Interrupt level 0 (highest) EI$^2$OS
                                ;Channel 0
;Data bank register (DTB) = 00H
       MOV   BAPL,#00H          ;Address for storing output data set
       MOV   BAPM,#06H          ;($600_H$ to $60A_H$ used)
       MOV   BAPH,#00H
       MOV   ISCS,#12H          ;Byte transfer, buffer address +1,
                                ;I/O address fixed,
                                ;transfer from memory to I/O
       MOV   IOAL,#00H          ;Set port 0 as transfer destination
       MOV   IOAH,#00H          ;address pointer
       MOV   DCTL,#0AH          ;Set transfer count to 10
       MOV   DCTH,#00H
;
       CLRB  I:EN8              ;INT8 disabled using ENIR1
       MOV   I:ELVR1L,#00000001B;H level detection set for INT8
       CLRB  I:ER8              ;INT8 interrupt request flag cleared
                                ;using EIRR1
       SETB  I:EN8              ;INT8 interrupt request enabled using
                                 ENIR1
       MOV   ILM,#07H           ;Set ILM in PS to level 7
       OR    CCR,#40H           ;Interrupts enabled
LOOP:
       ÅE
       Processing by user
       ÅE
       BRA   LOOP
;---------Interrupt program------------------------------------
WARI:
       CLRB   I:ER8              ;INT8 interrupt request flag cleared
       ÅE
       Processing by user
       ÅE
        RETI                    ;Return from interrupt processing
CODE   ENDS
;---------Vector setting----------------------------------------
VECT   CSEG  ABS=0FFH
       ORG   00FF94H            ;Set vector to interrupt number
```

```
                                 #26(1AH)
       DSL   WARI
       ORG   00FFDCH            ;Reset vector set
       DSL   START
       DB    00H                ;Set to single-chip mode
VECT  ENDS
      END    START
```

CHAPTER 18

8-/10-BIT A/D CONVERTER

This chapter explains the functions and operation of 8-/10-bit A/D converter.

18.1 Overview of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter converts the analog input voltage to a 8- or 10-bit digital value by using the RC sequential-comparison converter system.

- **An input signal can be selected from the input signals of the analog input pins for 16 channels.**
- **The start trigger can be selected from a software trigger and an external trigger.**

■ Function of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter converts the analog voltage (input voltage) input to the analog input pin into an 8- or 10-bit digital value (A/D conversion).

The 8-/10-bit A/D converter has the following functions:

- A/D conversion time is a minimum of 1.9 μs* per channel including sampling time.
- Sampling time is a minimum of 0.5 μs* per channel.
- RC sequential-comparison converter system with sample & hold circuit
- Setting of 8-bit or 10-bit resolution enabled
- Analog input pin can be used up to 16 channels.
- Generates interrupt request by storing A/D conversion results in A/D data register
- Starts EI^{2}OS if interrupt request generated. Use of the EI^{2}OS prevents data loss even at continuous A/D conversion.
- Selects start trigger from software trigger and external trigger (falling edge)

*: When the machine clock frequency operates at 24 MHz and $AV_{CC} \geq 4.5$ V.

■ Conversion Modes of 8-/10-bit A/D Converter

There are 3 conversion modes of 8-/10-bit A/D converter as shown below:

Table 18.1-1 Conversion Modes of 8-/10-bit A/D Converter

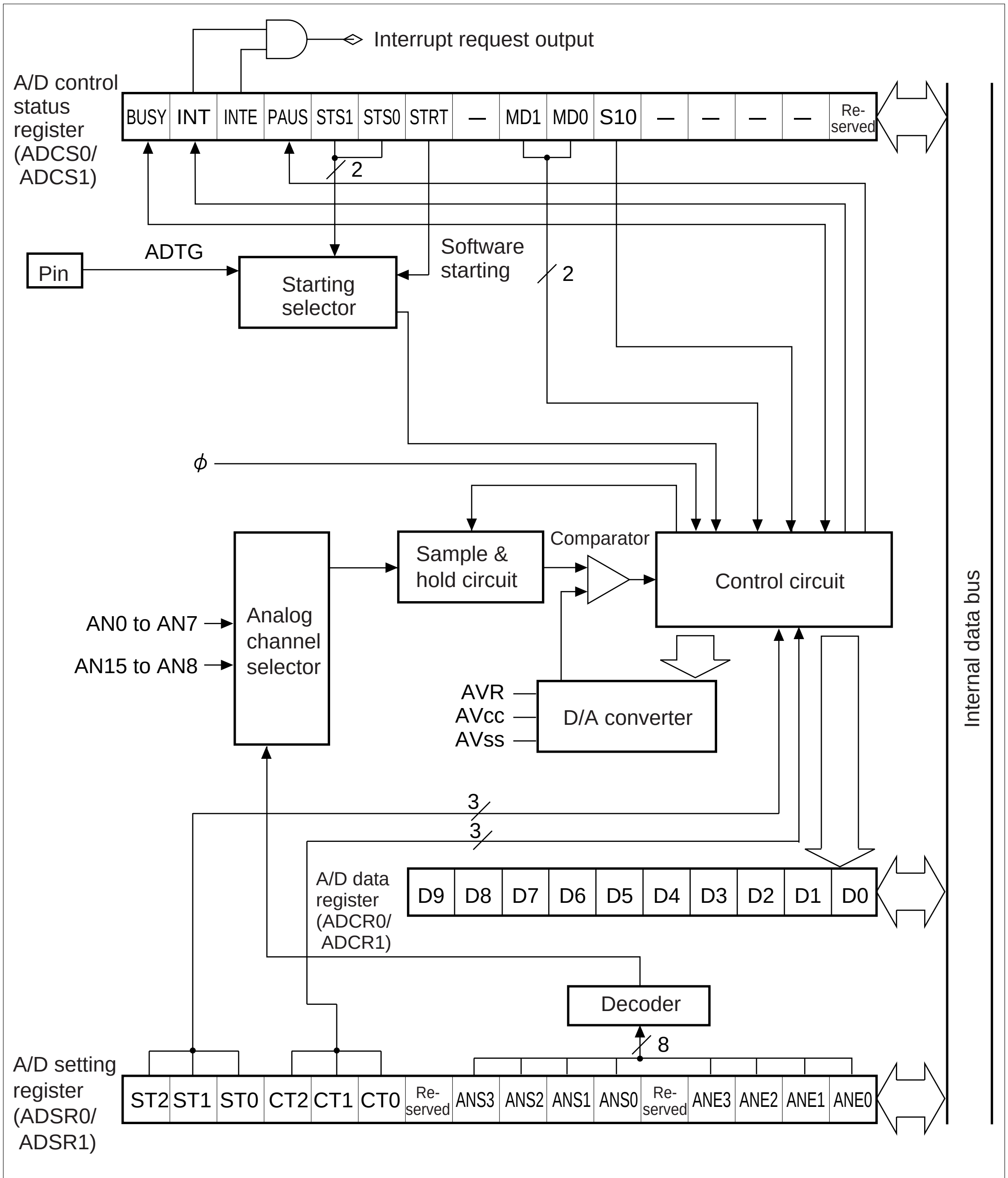
Conversion Mode	Description
Single-shot conversion mode	A/D conversion is performed sequentially from the start channel to the end channel. When A/D conversion for the end channel is terminated, it stops.
Continuous conversion mode	A/D conversion is performed sequentially from the start channel to the end channel. When A/D conversion for the end channel is terminated, it is continued after returning to the start channel.
Pause-conversion mode	A/D conversion is performed pausing per channel. When A/D conversion for the end channel is terminated, A/D conversion and pause are repeated after returning to the start channel.

18.2 Block Diagram of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter consists of following blocks.

■ Block Diagram of 8-/10-bit A/D Converter

Figure 18.2-1 Block Diagram of 8-/10-bit A/D Converter



—	: Undefined
Reserved	: Always set to 0.
ϕ	: Machine clock

- Details of pins in block diagram

Table 18.2-1 shows the actual pin names and interrupt request numbers of the 8-/10-bit A/D converter.

Table 18.2-1 Pins and Interrupt Request Numbers in Block Diagram

Pin Name/Interrupt Request Number in Block Diagram		Actual Pin Name/Interrupt Request Number
ADTG	Trigger input pin	P80/ADTG/INT12R
AN0 to AN7	Analog input pin ch0 to ch7	P60/AN0 to P65/AN5 P66/AN6/PPGC(D) P67/AN7/PPGE(F)
AN8 to AN15	Analog input pin ch8 to ch15	P50/AN8 to P52/AN10 P53/AN11/TIN3 P54/AN12/TOT3/INT8 P55/AN13/INT10 P56/AN14/INT11 P57/AN15/INT13
AVR	Vref+ input pin	AVR
AV_{CC}	V_{CC} input pin	AV_{CC}
AV_{SS}	V_{SS} input pin	AV_{SS}
Interrupt request output	Interrupt request output	#29($1D_H$)

- A/D control status registers (ADCS)

This register starts the A/D conversion function by software, selects the start trigger for the A/D conversion function, selects the conversion mode, enables or disables an interrupt request, checks and clears the interrupt request flag, temporarily stops A/D conversion and checks the state during conversion, and sets the resolution.

- A/D data registers (ADCR)

This register stores the A/D conversion results.

- A/D setting register (ADSR)

Starting channel and end channel of A/D conversion, compare time of A/D conversion and sampling time are set.

- Start selector

This selector selects the trigger to start A/D conversion. An external pin input can be set as the start trigger.

- Decoder

This decoder sets the A/D conversion start channel select bits and the A/D conversion end channel select bits in the A/D control status register (ADSR0:ANS3 to ANS0 and ANE3 to ANE0) to select the analog input pin to be used for A/D conversion.

● Analog channel selector

This selector selects the pin to be used for A/D conversion from the 16-channel analog input pins by receiving a signal from the decoder.

● Sample & hold circuit

This circuit holds the input voltage selected by the analog channel selector. By holding the input voltage immediately after A/D conversion is started, A/D conversion is performed without being affected by the fluctuation of the input voltage during A/D conversion.

● D/A converter

This converter generates the reference voltage which is compared with the input voltage held in the sample & hold circuit.

● Comparator

This comparator compares the D/A converter output voltage with input voltage held in the sample & hold circuit to determine the amount of voltage.

● Controller

This circuit determines the A/D conversion value by receiving the signal indicating the amount of voltage determined by the comparator. When the A/D conversion results are determined, the result data is stored in the A/D data register. If an interrupt request is enabled, an interrupt is generated.

18.3 Configuration of 8-/10-bit A/D Converter

This section explains the pins, registers, and interrupt factors of the A/D converter.

■ Pins of 8-/10-bit A/D Converter

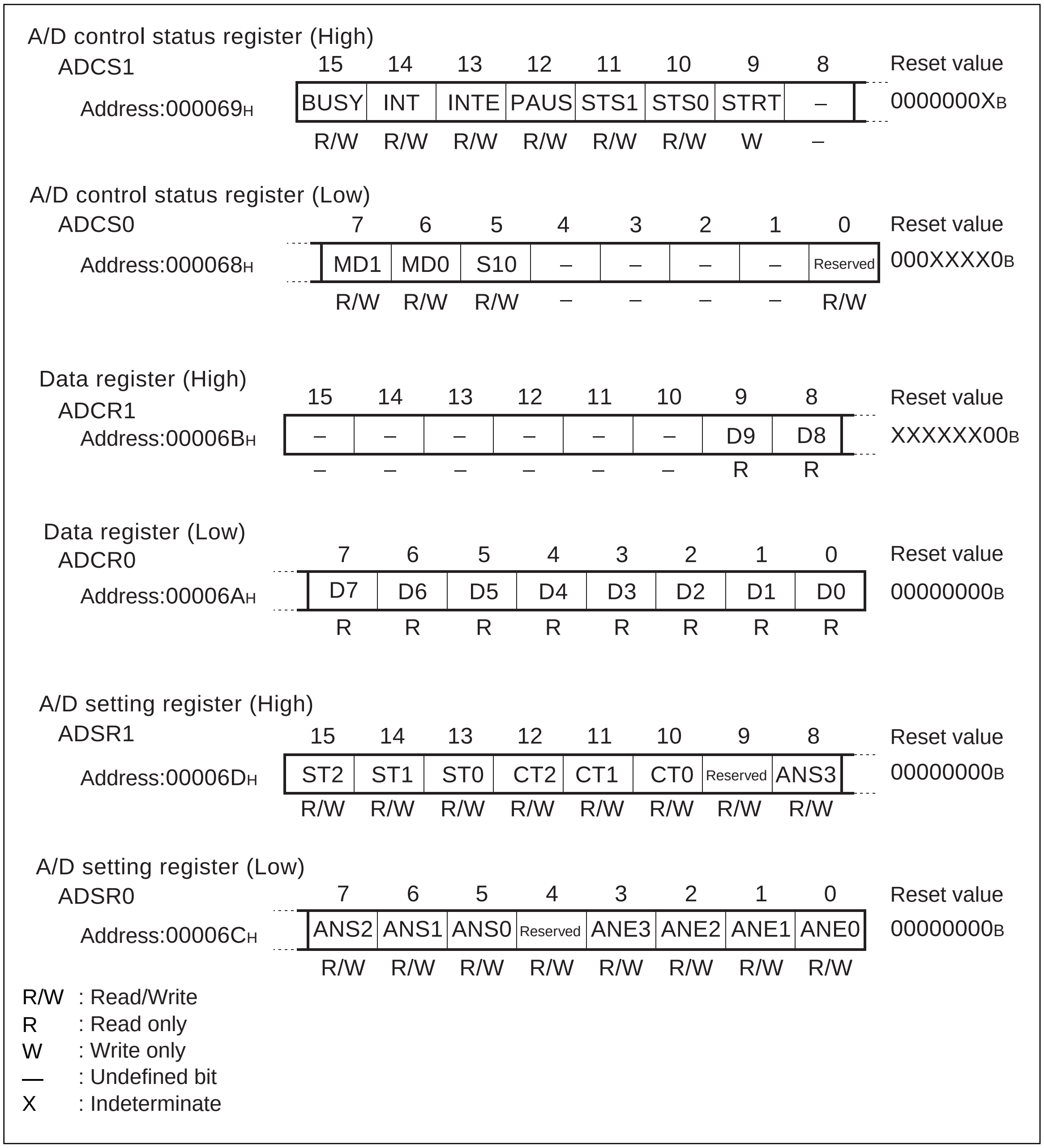
The pins of the 8-/10-bit A/D converter serve as general-purpose I/O ports. Table 18.3-1 shows the pin functions and the setting required for use of the 8-/10-bit A/D converter.

Table 18.3-1 Pins of 8-/10-bit A/D Converter

Function Name	Pin Name	Pin Function	Setting Required for Use of 8-/10-bit A/D Converter
Trigger input	P80 / ADTG/INT12R	General-purpose I/O port, external trigger input, external interrupt	Set as input port in port direction register (DDR8).
Channel 0	P60 / AN0	General-purpose I/O ports, analog inputs	Enable input of analog signal (ADER6: set the corresponding bit of ADE7 to ADE0 to "1")
Channel 1	P61 / AN1		
Channel 2	P62 / AN2		
Channel 3	P63 / AN3		
Channel 4	P64 / AN4		
Channel 5	P65 / AN5		
Channel 6	P66 / AN6/PPGC(D)	General-purpose I/O ports, analog inputs, PPG output	
Channel 7	P67 / AN7/PPGE(F)	General-purpose I/O ports, analog inputs, PPG output	
Channel 8	P50 / AN8	General-purpose I/O ports, analog inputs	Enable input of analog signal (ADER5: set the corresponding bit of ADE15 to ADE8 to "1")
Channel 9	P51 / AN9		
Channel 10	P52 / AN10		
Channel 11	P53 / AN11/TIN3	General-purpose I/O ports, analog inputs, event input pin for reload timer	
Channel 12	P54 / AN12/TOT3/ INT8	General-purpose I/O ports, analog inputs, external interrupt input, output pin for reload timer	
Channel 13	P55 / AN13/INT10	General-purpose I/O ports, analog inputs, external interrupt input	
Channel 14	P56 / AN14/INT11		
Channel 15	P57 / AN15/INT13		

■ List of Registers and Reset Values of 8-/10-bit A/D Converter

Figure 18.3-1 List of Register and Reset Value of 8-/10-bit A/D Converter



A/D control status register (High)
ADCS1
Address:000069H

15	14	13	12	11	10	9	8	Reset value
BUSY	INT	INTE	PAUS	STS1	STS0	STRT	–	0000000XB
R/W	R/W	R/W	R/W	R/W	R/W	W	–	

A/D control status register (Low)
ADCS0
Address:000068H

7	6	5	4	3	2	1	0	Reset value
MD1	MD0	S10	–	–	–	–	Reserved	000XXXX0B
R/W	R/W	R/W	–	–	–	–	R/W	

Data register (High)
ADCR1
Address:00006BH

15	14	13	12	11	10	9	8	Reset value
–	–	–	–	–	–	D9	D8	XXXXXX00B
–	–	–	–	–	–	R	R	

Data register (Low)
ADCR0
Address:00006AH

7	6	5	4	3	2	1	0	Reset value
D7	D6	D5	D4	D3	D2	D1	D0	00000000B
R	R	R	R	R	R	R	R	

A/D setting register (High)
ADSR1
Address:00006DH

15	14	13	12	11	10	9	8	Reset value
ST2	ST1	ST0	CT2	CT1	CT0	Reserved	ANS3	00000000B
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

A/D setting register (Low)
ADSR0
Address:00006CH

7	6	5	4	3	2	1	0	Reset value
ANS2	ANS1	ANS0	Reserved	ANE3	ANE2	ANE1	ANE0	00000000B
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W	: Read/Write
R	: Read only
W	: Write only
—	: Undefined bit
X	: Indeterminate

■ Generation of Interrupt from 8-/10-bit A/D Converter

In the 8-/10-bit A/D converter, when the A/D conversion results are stored in the A/D data register (ADCR0, 1), the interrupt request flag bit in the A/D control status register (ADCS1:INT) is set to "1". When an interrupt request is enabled (ADCS1:INTE = 1), an interrupt is generated.

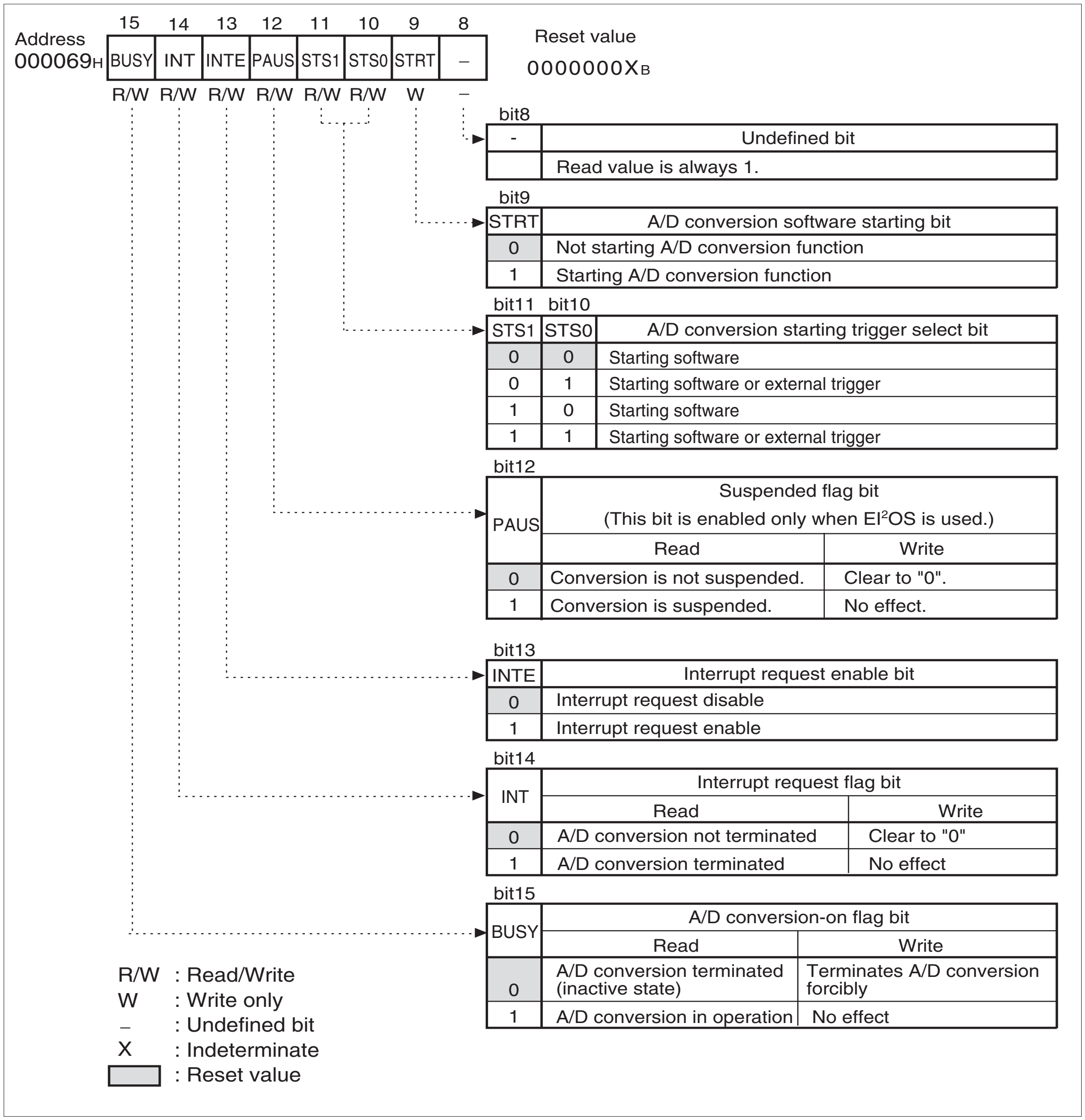
18.3.1 A/D Control Status Register (High) (ADCS1)

The A/D control status register (High) (ADCS1) provides the following settings:

- **Starting A/D conversion function by software**
- **Selecting start trigger for A/D conversion**
- **Storing A/D conversion results in A/D data register to enable or disable interrupt request**
- **Storing A/D conversion results in A/D data register to check and clear interrupt request flag**
- **Pausing A/D conversion and checking state during conversion**

■ A/D Control Status Register (High) (ADCS1)

Figure 18.3-2 A/D Control Status Register (High) (ADCS1)



Address	15	14	13	12	11	10	9	8	Reset value
000069H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	–	0000000XB
	R/W	R/W	R/W	R/W	R/W	R/W	W	–	

bit8

-	Undefined bit
	Read value is always 1.

bit9

STRT	A/D conversion software starting bit
0	Not starting A/D conversion function
1	Starting A/D conversion function

bit11	bit10	
STS1	STS0	A/D conversion starting trigger select bit
0	0	Starting software
0	1	Starting software or external trigger
1	0	Starting software
1	1	Starting software or external trigger

bit12

PAUS	Suspended flag bit (This bit is enabled only when EI²OS is used.)	
	Read	Write
0	Conversion is not suspended.	Clear to "0".
1	Conversion is suspended.	No effect.

bit13

INTE	Interrupt request enable bit
0	Interrupt request disable
1	Interrupt request enable

bit14

INT	Interrupt request flag bit	
	Read	Write
0	A/D conversion not terminated	Clear to "0"
1	A/D conversion terminated	No effect

bit15

BUSY	A/D conversion-on flag bit	
	Read	Write
0	A/D conversion terminated (inactive state)	Terminates A/D conversion forcibly
1	A/D conversion in operation	No effect

R/W	: Read/Write
W	: Write only
–	: Undefined bit
X	: Indeterminate
▨	: Reset value

Table 18.3-2 Function of Each Bit of A/D Control Status Register (High) (ADCS1) (1/2)

Bit name		Function
bit15	BUSY: A/D conversion-on flag bit	This bit forcibly terminates the 8-/10-bit A/D converter. When read, this bit indicates whether the 8-/10-bit A/D converter is operating or stopped. **When set to "0":** Forcibly terminates 8-/10-bit A/D converter **When set to "1":** No effect **Read:** 1 is read when the 8-/10-bit A/D converter is operating and 0 is read when the converter is stopped. **Note:** • "1" is read from this bit when an read-modify-write instruction is used. • In the single-shot mode, this bit is cleared when A/D conversion ends. • In the continuous or pause mode, the A/D conversion does not stop until writing "0" to this bit. • Do not perform the forced stop (BUSY=0) and the activation of the A/D converter concurrently (using software (STRT=1), external trigger, or timer).
bit14	INT: Interrupt request flag bit	This bit indicates that an interrupt request is generated. • When A/D conversion is terminated and its results are stored in the A/D data register (ADCR0, 1), the INT bit is set to "1". • When the interrupt request flag bit is set (INT = 1) with an interrupt request enabled (INTE = 1), an interrupt request is generated. **When set to "0":** Cleared **When set to "1":** No effect When EI^{2}OS function started: Cleared **Note:** • "1" is read from this bit when an read-modify-write instruction is used. • To clear the INT bit, write 0 when the 8-/10-bit A/D converter is stopped.
bit13	INTE: Interrupt request enable bit	This bit enables or disables output of an interrupt request. • When the interrupt request flag bit is set (INT =1) with an interrupt request enabled (INTE = 1), an interrupt request is generated. Note: Always set this bit to 1 when the EI^{2}OS function is used.

Table 18.3-2 Function of Each Bit of A/D Control Status Register (High) (ADCS1) (2/2)

Bit name		Function
bit12	PAUS: Pause flag bit	This bit indicates the A/D conversion operating state when the EI^{2}OS function is used. • The PAUS bit is enabled only when the EI^{2}OS function is used. • When next A/D conversion terminates before the A/D conversion result completes the transfer from the A/D data register (ADCR0, 1) to memory, the A/D conversion pauses in order to prevent previous data from being overwritten. When the A/D conversion pauses, the PAUS bit is set to "1". • After transfer of the A/D conversion results to memory, the 8-/10-bit A/D converter automatically resumes A/D conversion. **Note:** • See "18.5.5 A/D-converted Data Protection Function" for the conversion data protection function. • This bit is not cleared even if the pause state is cancelled. To clear this bit, write "0" to it.
bit11, bit10	STS1, STS0: A/D conversion start trigger select bits	These bits select the trigger to start the 8-/10-bit A/D converter. • 00_B: Software start • 01_B: External pin trigger or software start • 10_B: Software start • 11_B: External pin trigger or software start **Note:** • When the falling edge is detected in the ADTG pin at selected external terminal trigger (01_B, 11_B), the A/D conversion is begun. • If two or more start triggers are set (other than STS1, STS0="00_B", "10_B"), the 8-/10-bit A/D converter is started by the first-generated start trigger. • Start trigger setting should be changed when the operation of resource generating a start trigger is stopped (trigger is inactive).
bit9	STRT: A/D conversion software start bit	This bits starts the 8-/10-bit A/D converter by software. **When set to "1":** Starts 8-/10-bit A/D converter • If A/D conversion pauses in the pause-conversion mode, it is resumed by writing 1 to the STRT bit. **When set to "0":** Invalid. The state remains unchanged. Note: • The read-modify-write instructions read "0". • The byte/word command reads "1". • Do not perform forcible termination (BUSY = 0) and software start (STRT = 1) of the 8-/10-bit A/D converter simultaneously.
bit8	Undefined bit	• Read: "1" is always read. • Write: No effect

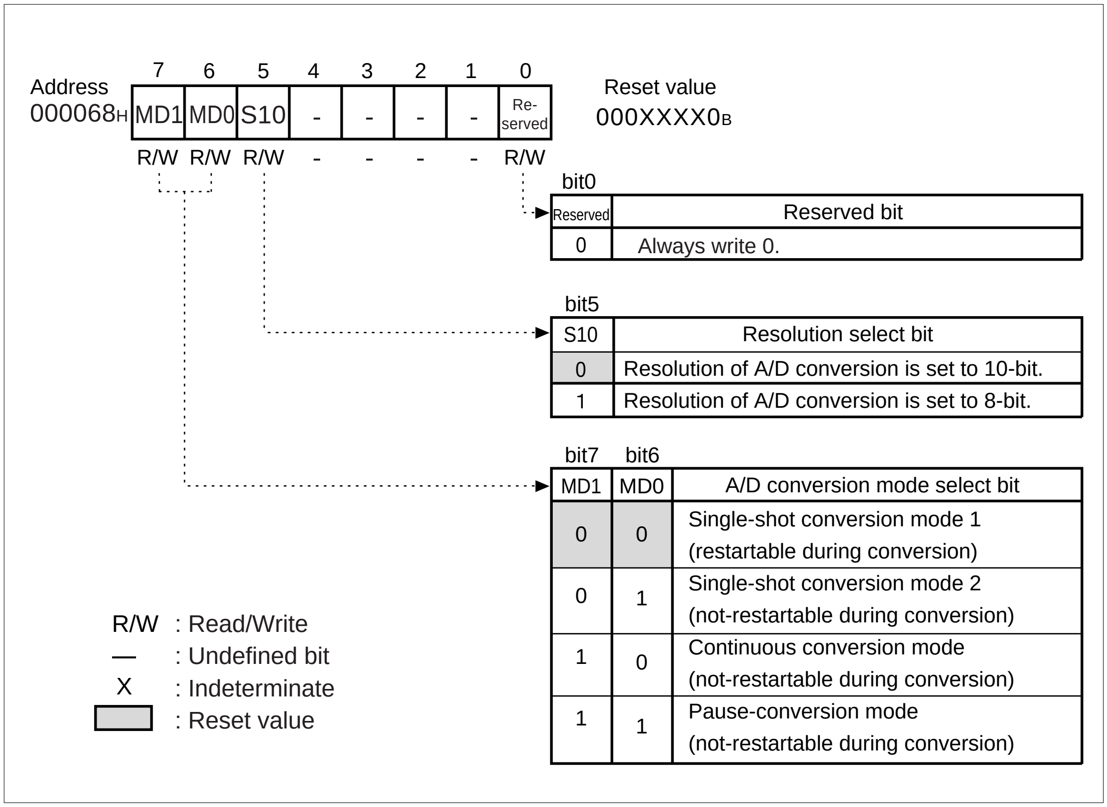
18.3.2 A/D Control Status Register (Low) (ADCS0)

The A/D control status register (Low) (ADCS0) provides the following settings:

- **Selecting A/D conversion mode**
- **Selecting start channel and end channel of A/D conversion**

■ A/D Control Status Register (Low) (ADCS0)

Figure 18.3-3 A/D Control Status Register (Low) (ADCS0)



Reserved	Reserved bit
0	Always write 0.

S10	Resolution select bit
▨ 0	Resolution of A/D conversion is set to 10-bit.
1	Resolution of A/D conversion is set to 8-bit.

MD1	MD0	A/D conversion mode select bit
▨ 0	▨ 0	Single-shot conversion mode 1 (restartable during conversion)
0	1	Single-shot conversion mode 2 (not-restartable during conversion)
1	0	Continuous conversion mode (not-restartable during conversion)
1	1	Pause-conversion mode (not-restartable during conversion)

R/W	: Read/Write
—	: Undefined bit
X	: Indeterminate
▨	: Reset value

Table 18.3-3 Function of Each Bit of A/D Control Status Register (Low) (ADCS0)

Bit Name		Function
bit7 bit6	MD1, MD0: A/D conversion mode select bits	These bits set the A/D conversion mode. **Single-shot conversion mode 1:** • The analog inputs from the start channel (ADSR0 : ANS3 to ANS0) to the end channel (ADSR0 : ANE3 to ANE0) are A/D-converted continuously. • The A/D conversion pauses after A/D conversion for the end channel. • This mode can be restarted during A/D conversion. **Single-shot conversion mode 2:** • The analog inputs from the start channel (ADSR0 : ANS3 to ANS0) to the end channel (ADSR0 : ANE3 to ANE0) are A/D-converted continuously. • The A/D conversion pauses after A/D conversion for the end channel. • This mode cannot be restarted during A/D conversion. **Continuous conversion mode:** • The analog inputs from the start channel (ADSR0 : ANS3 to ANS0) to the end channel (ADSR0 : ANE3 to ANE0) are A/D-converted continuously. • When A/D conversion for the end channel is terminated, it is continued after returning to the analog input for the start channel. • To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS1:BUSY). • This mode cannot be restarted during A/D conversion. **Pause conversion mode:** • A/D conversion for the start channel (ADSR0 : ANS3 to ANS0) starts. The A/D conversion pauses at termination of A/D conversion for a channel. When the start trigger is inputted while A/D conversion pauses, A/D conversion for the next channel is started. • The A/D conversion pauses at the termination of A/D conversion for the end channel. When the start trigger is inputted while A/D conversion pauses, A/D conversion is continued after returning to the analog input for the start channel. • To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS1:BUSY). • This mode cannot be restarted during A/D conversion. **Note:** • To change the conversion mode, perform at the stop state before starting the A/D conversion. • When the conversion mode is set to "not restartable" (other than MD1, MD0 ="00_B"), it cannot be restarted with any start triggers (software trigger and external trigger) during A/D conversion.
bit5	S10: Resolution select bit	This bit sets the resolution of the A/D conversion. **When set to "0":** Set the resolution of the A/D conversion to 10-bit of A/D conversion data bits D9 to D0. **When set to "1":** Set the resolution of the A/D conversion to 8-bit of A/D conversion data bits D7 to D0. Note: To change the S10 bit, perform at the stop state before starting the A/D conversion. When the S10 bit is changed after starting A/D conversion, the converted result stored in the A/D conversion data bit (D9 to D0) is invalid.

18.3.3 A/D Data Register (ADCR0/ADCR1)

The A/D data register (ADCR0/ADCR1) stores the digital value generated as the conversion result. The ADCR0 stores the lower 8-bit, and ADCR1 stores the most significant 2-bit of the conversion result. This register is rewritten each time the conversion complete and stores last conversion value normally.

■ A/D Data Register (ADCR0/ADCR1)

Figure 18.3-4 A/D Data Register (ADCR0/ADCR1)

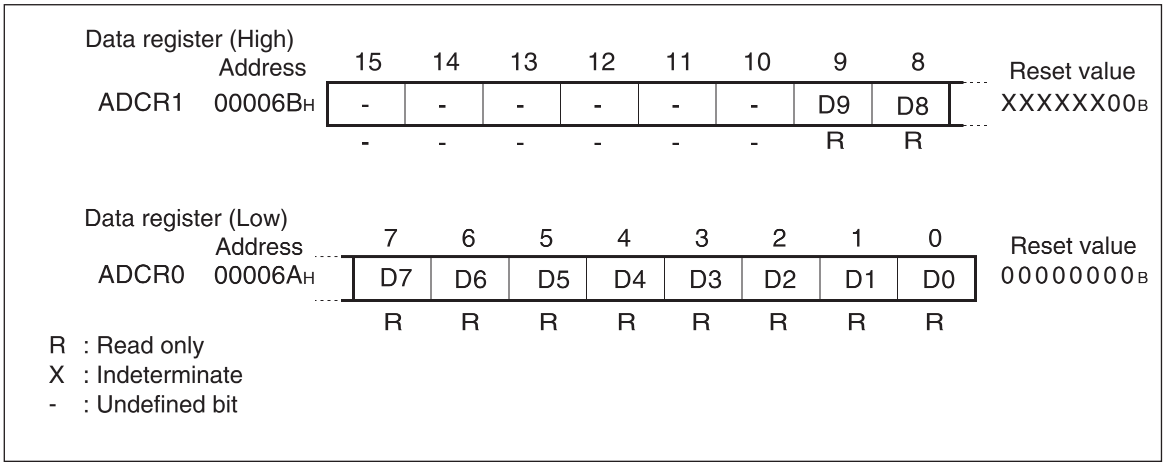


Table 18.3-4 Functions of A/D Data Register (ADCR0/ADCR1)

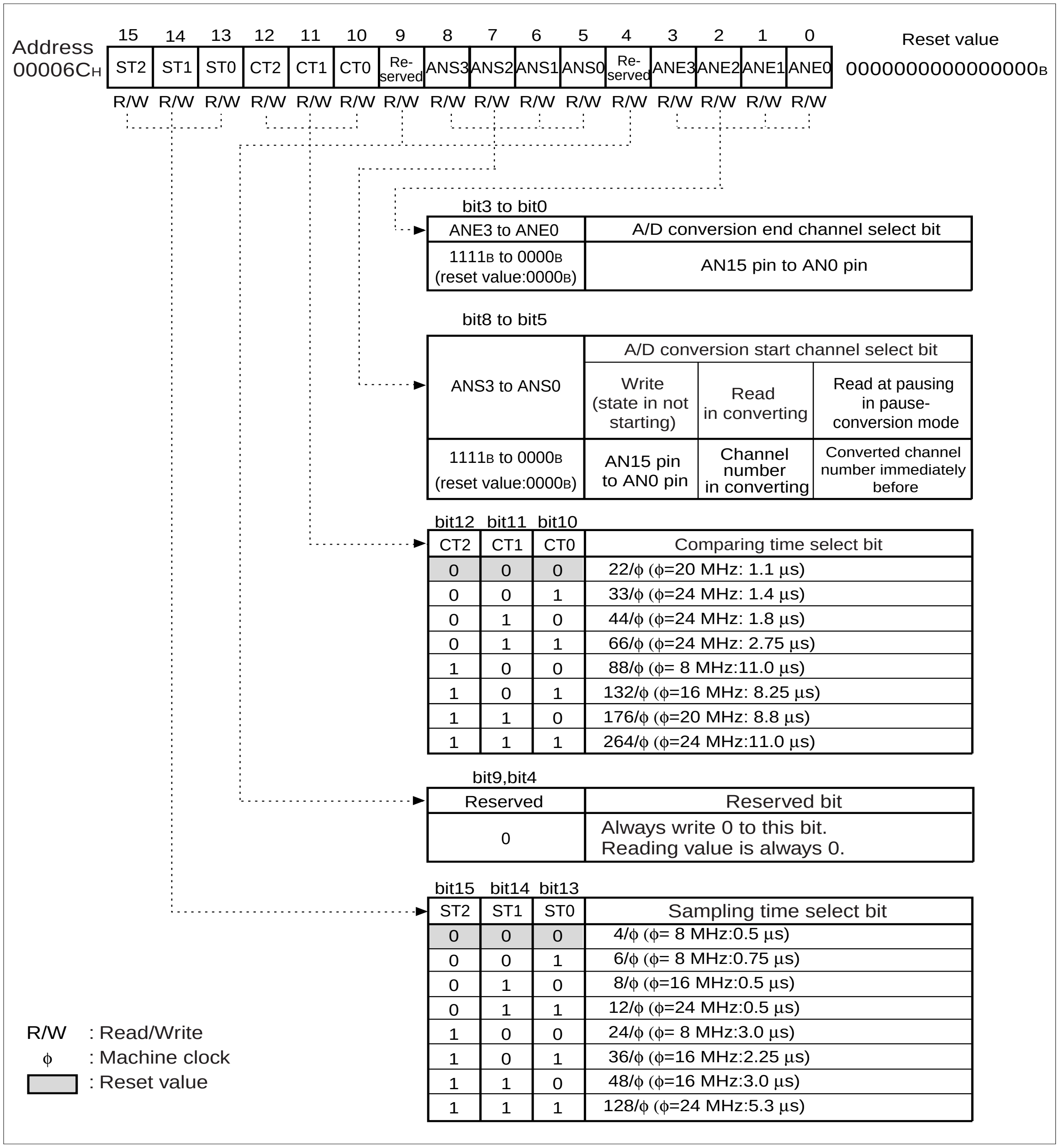
Bit Name		Function
bit15 to bit10	Undefined bits	1 is always read at reading.
bit9 to bit0	D9 to D0: A/D conversion data bits	These bits store the A/D conversion results. **When resolution set in 10 bits (S10=0):** Conversion data is stored in the 10 bits from D9 to D0. **When resolution set in 8 bits (S10=1):** Conversion data is stored in the 8 bits from D7 to D0. In this case, the read values of D9 and D8 are 1. **Note:** • Writing to this register is disabled. • Use a word instruction (MOVW) to read the A/D conversion results stored in the A/D conversion data bits (D9 to D0).

18.3.4 A/D Setting Register (ADSR0/ADSR1)

A/D setting register (ADSR0/ADSR1) can set as following.
- **Setting of A/D conversion time (sampling time and comparing time)**
- **Setting of sampling channel (starting channel and end channel)**
- **Displaying the present sampling channels**

■ A/D Setting Register (ADSR0/ADSR1)

Figure 18.3-5 A/D Setting Register (ADSR0/ADSR1)



Address 00006CH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST2	ST1	ST0	CT2	CT1	CT0	Re-served	ANS3	ANS2	ANS1	ANS0	Re-served	ANE3	ANE2	ANE1	ANE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Reset value 0000000000000000B

bit3 to bit0

ANE3 to ANE0	A/D conversion end channel select bit
1111B to 0000B (reset value:0000B)	AN15 pin to AN0 pin

bit8 to bit5

ANS3 to ANS0	A/D conversion start channel select bit		
	Write (state in not starting)	Read in converting	Read at pausing in pause-conversion mode
1111B to 0000B (reset value:0000B)	AN15 pin to AN0 pin	Channel number in converting	Converted channel number immediately before

bit12 CT2	bit11 CT1	bit10 CT0	Comparing time select bit
0	0	0	22/ϕ (ϕ=20 MHz: 1.1 µs)
0	0	1	33/ϕ (ϕ=24 MHz: 1.4 µs)
0	1	0	44/ϕ (ϕ=24 MHz: 1.8 µs)
0	1	1	66/ϕ (ϕ=24 MHz: 2.75 µs)
1	0	0	88/ϕ (ϕ= 8 MHz:11.0 µs)
1	0	1	132/ϕ (ϕ=16 MHz: 8.25 µs)
1	1	0	176/ϕ (ϕ=20 MHz: 8.8 µs)
1	1	1	264/ϕ (ϕ=24 MHz:11.0 µs)

bit9,bit4

Reserved	Reserved bit
0	Always write 0 to this bit. Reading value is always 0.

bit15 ST2	bit14 ST1	bit13 ST0	Sampling time select bit
0	0	0	4/ϕ (ϕ= 8 MHz:0.5 µs)
0	0	1	6/ϕ (ϕ= 8 MHz:0.75 µs)
0	1	0	8/ϕ (ϕ=16 MHz:0.5 µs)
0	1	1	12/ϕ (ϕ=24 MHz:0.5 µs)
1	0	0	24/ϕ (ϕ= 8 MHz:3.0 µs)
1	0	1	36/ϕ (ϕ=16 MHz:2.25 µs)
1	1	0	48/ϕ (ϕ=16 MHz:3.0 µs)
1	1	1	128/ϕ (ϕ=24 MHz:5.3 µs)

R/W : Read/Write
ϕ : Machine clock
▨ : Reset value

Table 18.3-5 Function of A/D Setting Register (ADSR0/ADSR1) (1/2)

Bit Name		Function
bit15 to bit13	ST2, ST1, ST0: Sampling time select bits	These bits set the sampling time of A/D conversion. • These bits set the time when the A/D conversion is started and inputted analog voltage is sampled by the sample & hold circuit until it is retained. • See Table 18.3-6 for the setting of these bits.
bit12 to bit10	CT2, CT1, CT0: Comparing time select bits	Comparing time of A/D conversion (comparing time) is set. • These bits set the time when the analog input is A/D converted until it stores to the data bits (D9 to D0). • See Table 18.3-7 for the setting of these bits.
bit9, bit4	Reserved bits	Always write 0 to these bits. Reading value is always 0.
bit8 to bit5	ANS3 to ANS0: A/D conversion start channel select bits	These bits set the channel at which A/D conversion start.At read, the channel number under conversion can be checked if the A/D conversion is in progress and last A/D converted channel number can be checked if the A/D conversion is completed or is stopping. And before A/D conversion starts, the previous conversion channel will be read even if these bits have already been set to the new value. These bits are initialized to "0000_B" at reset. **Start channel < end channel:** A/D conversion starts at channel set by A/D conversion start channel select bits (ANS3 to ANS0) and terminates channel set by A/D conversion end channel select bits (ANE3 to ANE0) **Start channel = end channel:** A/D conversion is performed only for one channel set by A/D conversion start (= end) channel select bits (ANS3 to ANS0 = ANE3 to ANE0) **Start channel > end channel:** Do not set **Continuous conversion mode and pause-conversion mode:** When A/D conversion terminated at the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), it returns to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0). **Read (Other than pause-conversion mode) :** The channel numbers (15 to 0) under A/D conversion are read. **Read (Pause-conversion mode) :** At read during a pause, the channel number A/D-converted immediately before a pause is read. **Notes:** • Do not set the A/D conversion start channel bits (ANS3 to ANS0) during A/D conversion. • Access in units of word when writing to these bits. If the byte write or bit operation is performed, the A/D conversion may be started from unexpected channel.

Table 18.3-5 Function of A/D Setting Register (ADSR0/ADSR1) (2/2)

Bit Name		Function
bit3 to bit0	ANE3 to ANE0: A/D conversion end channel select bits	These bits set the channel at which A/D conversion terminated. **Start channel < end channel:** A/D conversion starts at channel set by A/D conversion start channel select bits (ANS3 to ANS0) and terminates channel set by A/D conversion end channel select bits (ANE3 to ANE0) **Start channel = end channel:** A/D conversion is performed only for one channel set by A/D converter start (= end) channel select bits (ANE3 to ANE0 = ANS3 to ANS0). **Start channel > end channel:** Do not set. **Continuous conversion mode and pause-conversion mode:** When A/D conversion terminated at the channel set by the A/D conversion end channel select bits (ANE3 to ANE0), it returns to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0). Note: Do not set the A/D conversion end channel select bits (ANE3 to ANE0) during A/D conversion.

Note:

Do not set the A/D conversion mode set bits (MD1 and MD0) and A/D conversion end channel select bits (ANE3, ANE2, ANE1 and ANE0) through read-modify-write commands after the start channel is set in the A/D conversion start channel select bits (ANS3, ANS2, ANS1 and ANS0).
The ANS3, ANS2, ANS1 and ANS0 bits will read the last conversion channel until the A/D conversion operation starts. Accordingly when the MD1 and MD0 bits and the ANE3, ANE2, ANE1 and ANE0 bits are set through read-modify-write commands after the start channel is set in the ANS3, ANS2, ANS1 and ANS0 bits, the values of the ANE3, ANE2, ANE1 and ANE0 bits may be rewritten.

■ Setting of Sampling Time (ST2 to ST0 bits)

Table 18.3-6 Relation between ST2 to ST0 Bits and Sampling Time

ST2	ST1	ST0	Setting of Sampling Time	Setting example (φ: Internal operating frequency)
0	0	0	4 machine cycles	φ= 8 MHz: 0.5 µs
0	0	1	6 machine cycles	φ= 8 MHz: 0.75 µs
0	1	0	8 machine cycles	φ= 16 MHz: 0.5 µs
0	1	1	12 machine cycles	φ= 24 MHz: 0.5 µs
1	0	0	24 machine cycles	φ= 8 MHz: 3 µs
1	0	1	36 machine cycles	φ= 16 MHz: 2.25 µs
1	1	0	48 machine cycles	φ= 16 MHz: 3.0 µs
1	1	1	128 machine cycles	φ= 24 MHz: 5.3 µs

The sampling time must be set according to drive impedance R_{ext} connected to analog input. If the following condition is not met, the conversion accuracy will not be guaranteed.

- $R_{ext} \leq 1.5k\Omega$:
  - •- 4.5 V ≤ AV_{CC} < 5.5 V: The sampling time must be set greater than 0.5 µs.
  - • 4.0 V ≤ AV_{CC} < 4.5 V: The sampling time must be set greater than 1.2 µs.
- $R_{ext} > 1.5k\Omega$: The sampling time must be set greater than T_{samp} given by the following formula.
  - • 4.5 V ≤ AV_{CC} < 5.5 V: $T_{samp} = (2.52\ k\Omega + R_{ext}) \times 10.7\ pF \times 7$
  - • 4.0 V ≤ AV_{CC} < 4.5 V: $T_{samp} = (13.6\ k\Omega + R_{ext}) \times 10.7\ pF \times 7$

■ Setting of Comparing Time (CT2 to CT0 bits)

Table 18.3-7 Relation between CT2 to CT0 Bits and Comparing Time

CT2	CT1	CT0	Setting of comparing time	Setting example (ϕ: Internal operating frequency)
0	0	0	22 machine cycles	ϕ= 20 MHz: 1.1 µs
0	0	1	33 machine cycles	ϕ= 24 MHz: 1.4 µs
0	1	0	44 machine cycles	ϕ= 24 MHz: 1.8 µs
0	1	1	66 machine cycles	ϕ= 24 MHz: 2.75 µs
1	0	0	88 machine cycles	ϕ= 8 MHz: 11.0 µs
1	0	1	132 machine cycles	ϕ= 16 MHz: 8.25 µs
1	1	0	176 machine cycles	ϕ= 20 MHz: 8.8 µs
1	1	1	264 machine cycles	ϕ= 24 MHz: 11.0 µs

The comparing time must be set according to the analog power supply voltage AV_{CC}. If the following condition is not met, the conversion accuracy will not be guaranteed.

- 4.5 V ≤ AV_{CC} < 5.5 V: The comparing time must be set greater than 1.00 µs.
- 4.0 V ≤ AV_{CC} < 4.5 V: The comparing time must be set greater than 2.00 µs.

18.3.5 Analog Input Enable Register (ADER5, ADER6)

The analog input enable register enables or disables the analog input pins to be used in the 8-/10-bit A/D converter.

■ Analog Input Enable Register (ADER5, ADER 6)

Figure 18.3-6 Analog Input Enable Register (ADER5 to 6)

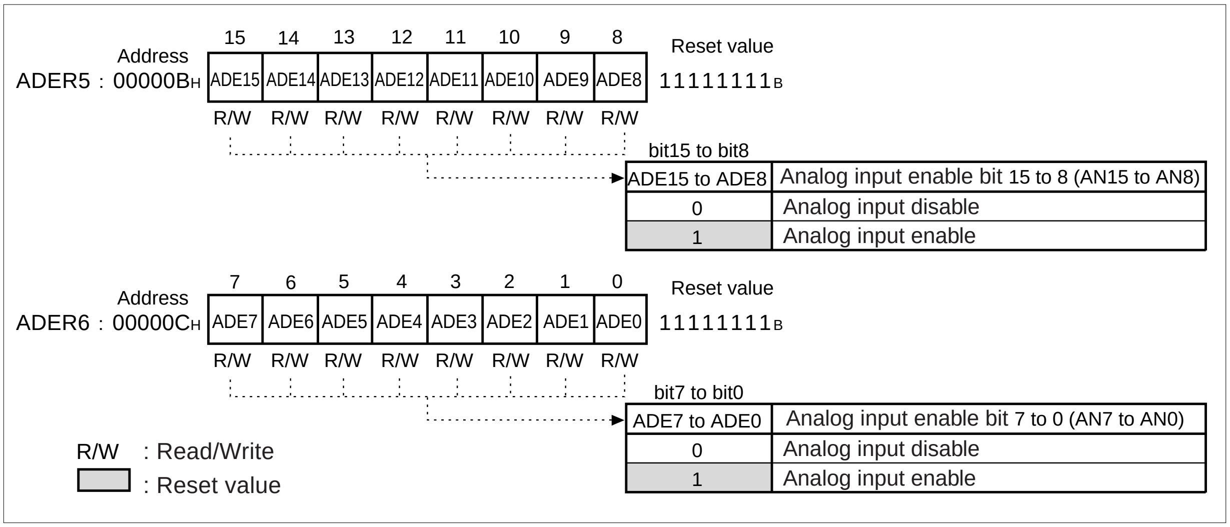


Table 18.3-8 Functions of Port 5 Analog Input Enable Register (ADER5)

Bit Name		Function
bit15 to bit8	ADE15 to ADE8: Analog input enable bits 15 to 8	These bits enable or disable the analog input pin (AN15 to AN8) of A/D conversion arranged on port 5. **When set to "0":** Disables analog input **When set to "1":** Enables analog input

Table 18.3-9 Functions of Port 6 Analog Input Enable Register (ADER6)

Bit Name		Function
bit0 to bit7	ADE7 to ADE0: Analog input enable bits 7 to 0	These bits enable or disable the analog input pin (AN7 to AN0) of A/D conversion arranged on port 6. **When set to "0":** Disables analog input **When set to "1":** Enables analog input

Notes:

- When using as the analog input pin, write "1" to the bit of the analog input enable register (ADER5, ADER6) corresponding to the pin to be used and set to the analog input.
- Setting the analog input pin to ADERx = "0" is disabled. Always set it to ADERx = "1".
- Each analog input pin serves as the general-purpose I/O port and I/O of peripheral function. The pin set to ADERx = "1" is forcibly set to the analog input pin regardless of the port direction register (DDR5, DDR6) and the I/O setting of each peripheral function.

18.4 Interrupt of 8-/10-bit A/D Converter

When A/D conversion is terminated and its results are stored in the A/D data register (ADCR), the 8-/10-bit A/D converter generates an interrupt request. The EI^{2}OS function can be used.

■ Interrupt of A/D Converter

When A/D conversion of the analog input voltage is terminated and its results are stored in the A/D data register (ADCR), the interrupt request flag bit in the A/D control status register (ADCS:INT) is set to "1". When the interrupt request flag bit is set (ADCS:INT = 1) with an interrupt request output enabled (ADCS:INTE = 1), an interrupt request is generated.

■ 8-/10-bit A/D Converter Interrupt and EI^{2}OS

Reference:	See "CHAPTER 3 INTERRUPTS" for details of the interrupt number, interrupt control register, and interrupt vector address.

■ EI^{2}OS Function of 8-/10-bit A/D Converter

In the 8-/10-bit A/D converter, the EI^{2}OS function can be used to transfer the A/D conversion results from the A/D data register (ADCR) to memory. If the EI^{2}OS function is used, the A/D-converted data protection function is activated to cause A/D conversion to pause during memory transfer and prevent data loss as A/D conversion is performed continuously.

18.5 Explanation of Operation of 8-/10-bit A/D Converter

The 8-/10-bit A/D converter has the following A/D conversion modes. Set each mode according to the setting of the A/D conversion mode select bits in the A/D control status register (ADCS:MD1, MD0).

- **Single-shot conversion mode (restartable/not-restartable during A/D conversion)**
- **Continuous conversion mode (not-restartable during A/D conversion)**
- **Pause--conversion mode (not-restartable during A/D conversion)**

■ Single-shot Conversion Mode (ADCS: MD1, MD0="00_B" or "01_B")

- When the start trigger is inputted, the analog inputs from the start channel (ADCS:ANS3 to ANS0) to the end channel (ADCS:ANE3 to ANE0) are A/D-converted continuously.
- The A/D conversion stops at the termination of the A/D conversion for the end channel.
- To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS:BUSY).
- When the A/D conversion mode select bits (MD1, MD0) are set to "00_B", this mode can be restarted during A/D conversion. If the bits are set to "01_B", this mode cannot be restarted during A/D conversion.

■ Continuous Conversion Mode (ADCS: MD1, MD0="10_B")

- When the start trigger is inputted, the analog inputs from the start channel (ADCS:ANS3 to ANS0) to the end channel (ADCS:ANE3 to ANE0) are A/D-converted continuously.
- When A/D conversion for the end channel is terminated, it is continued after returning to the analog input for the start channel.
- To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS:BUSY).
- This mode cannot be restarted during A/D conversion.

■ Pause-conversion Mode (ADCS: MD1, MD0="11_B")

- When the start trigger is inputted, A/D conversion starts for the start channel (ADCS:ANS3 to ANS0). The A/D conversion pauses at the termination of A/D conversion for one channel. When the start trigger is inputted while A/D conversion pauses, A/D conversion is performed for the next channel.
- The A/D conversion pauses at termination of A/D conversion for the end channel. When the start trigger is inputted while A/D conversion pauses, A/D conversion is continued after returning to the analog input for the start channel.
- To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS:BUSY).
- This mode cannot be restarted during A/D conversion.

18.5.1 Single-shot Conversion Mode

In the single-shot conversion mode, A/D conversion is performed sequentially from the start channel to the end channel. The A/D conversion stops at the termination of A/D conversion for the end channel.

■ Setting of Single-shot Conversion Mode

Operating the 8-/10-bit A/D converter in the single-shot conversion mode requires the setting shown in Figure 18.5-1 .

Figure 18.5-1 Setting of Single-shot Conversion Mode

	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
ADCS	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	–	MD1	MD0	S10	–	–	–	–	Re-served
	◎	◎	◎	◎	◎	◎	◎		0	◎	◎					0
ADCR	–	–	–	–	–	–	D9 to D0 (Converted data stored)									
ADSR	ST2	ST1	ST0	CT2	CT1	CT0	Re-served	ANS3	ANS2	ANS1	ANS0	Re-served	ANE3	ANE2	ANE1	ANE0
	◎	◎	◎	◎	◎	◎	0	◎	◎	◎	◎	0	◎	◎	◎	◎
ADER5	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8								
	○	○	○	○	○	○	○	○								
ADER6									ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0
									○	○	○	○	○	○	○	○

– : Undefined
◎ : Used bit
○ : Set the bit corresponding to pin to be used as analog input pin to 1.
0 : Setting 0

■ Operation of Single-shot Conversion Mode

- When the start trigger is inputted, A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) and is performed continuously up to the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).
- The A/D conversion stops at the termination of the A/D conversion for the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).
- To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS:BUSY).
- When the A/D conversion mode select bits (MD1, MD0) are set to "00_B", this mode can be restarted during A/D conversion. If the bits are set to "01_B", this mode cannot be restarted during A/D conversion.

[When start and end channels are the same]

If the start and end channels have the same channel number (ADCS: ANS3 to ANS0=ADCS: ANE3 to ANE0), only one A/D conversion for one channel set as the start channel (= end channel) is performed and terminated.

[Conversion order in single-shot conversion mode]

Table 18.5-1 gives an example of the conversion order in the single-shot conversion mode.

Table 18.5-1 Conversion Order in Single-shot Conversion Mode

Start Channel	End Channel	Conversion Order
AN0 pin (ADCS: ANS="0000_B")	AN3 pin (ADCS: ANE="0011_B")	AN0 → AN1 → AN2 → AN3 → end
AN3 pin (ADCS: ANS="0011_B")	AN3 pin (ADCS: ANE="0011_B")	AN3 → end

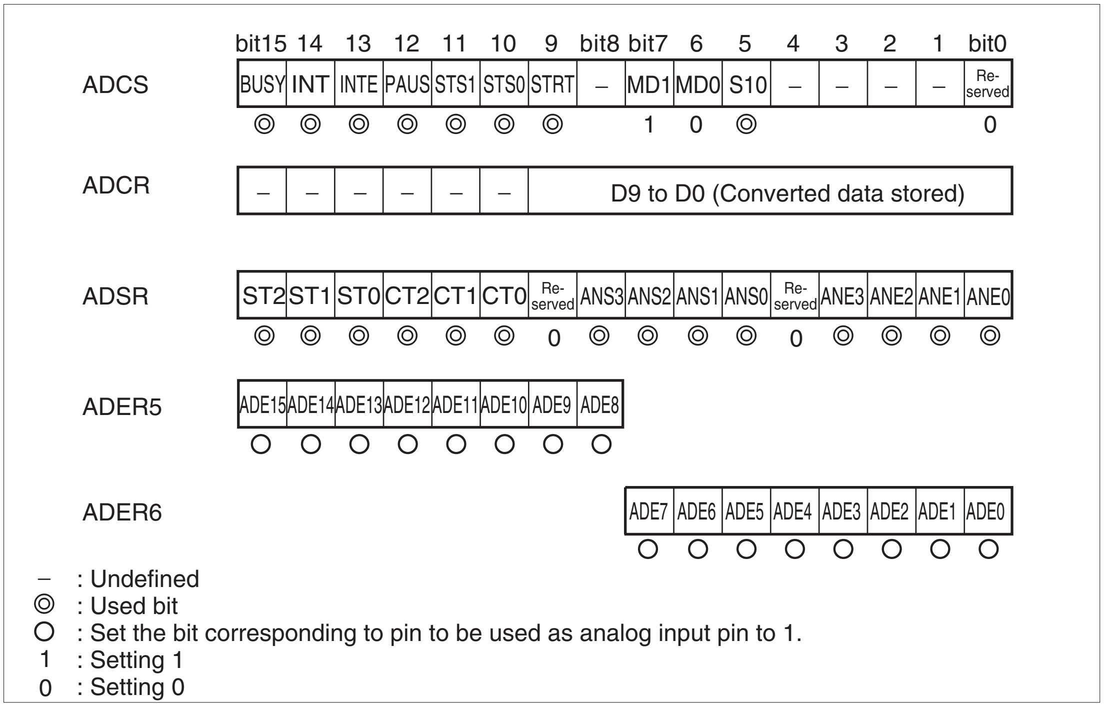
18.5.2 Continuous Conversion Mode

In the continuous conversion mode, A/D conversion is performed sequentially from the start channel to the end channel. When A/D conversion for the end channel is terminated, it is continued after returning to the start channel.

■ Setting of Continuous Conversion Mode

Operating the 8-/10-bit A/D converter in the continuous conversion mode requires the setting shown in Figure 18.5-2 .

Figure 18.5-2 Setting of Continuous Conversion Mode



	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
ADCS	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	–	MD1	MD0	S10	–	–	–	–	Re-served
	◎	◎	◎	◎	◎	◎	◎		1	0	◎					0
ADCR	–	–	–	–	–	–	D9 to D0 (Converted data stored)									
ADSR	ST2	ST1	ST0	CT2	CT1	CT0	Re-served	ANS3	ANS2	ANS1	ANS0	Re-served	ANE3	ANE2	ANE1	ANE0
	◎	◎	◎	◎	◎	◎	0	◎	◎	◎	◎	0	◎	◎	◎	◎
ADER5	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8								
	○	○	○	○	○	○	○	○								
ADER6									ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0
									○	○	○	○	○	○	○	○

– : Undefined
◎ : Used bit
○ : Set the bit corresponding to pin to be used as analog input pin to 1.
1 : Setting 1
0 : Setting 0

■ Operation of Continuous Conversion Mode

- When the start trigger is inputted, A/D conversion starts from the channel set by the A/D conversion start channel select bits (ANS3 to ANS0) and is performed continuously up to the channel set by the A/D conversion end channel select bits (ANE3 to ANE0).
- When A/D conversion for the channel set by the A/D conversion end channel select bits (ANE3 to ANE0) is terminated, it is continued after returning to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0).
- To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS:BUSY).
- This mode cannot be restarted during A/D conversion.

[When start and end channels are the same]

If the start and end channels have the same channel number (ADCS:ANS3 to ANS0 = ADCS:ANE3 to ANE0), A/D conversion for one channel set as the start channel (= end channel) is repeated.

[Conversion order in continuous conversion mode]

Table 18.5-2 gives an example of the conversion order in the continuous conversion mode.

Table 18.5-2 Conversion Order in Continuous Conversion Mode

Start Channel	End Channel	Conversion Order
AN0 pin (ADCS: ANS="0000_B")	AN3 pin (ADCS: ANE="0011_B")	AN0 → AN1 → AN2 → AN3 → AN0 → repeat
AN3 pin (ADCS: ANS="0011_B")	AN3 pin (ADCS: ANE="0011_B")	AN3 → AN3 → repeat

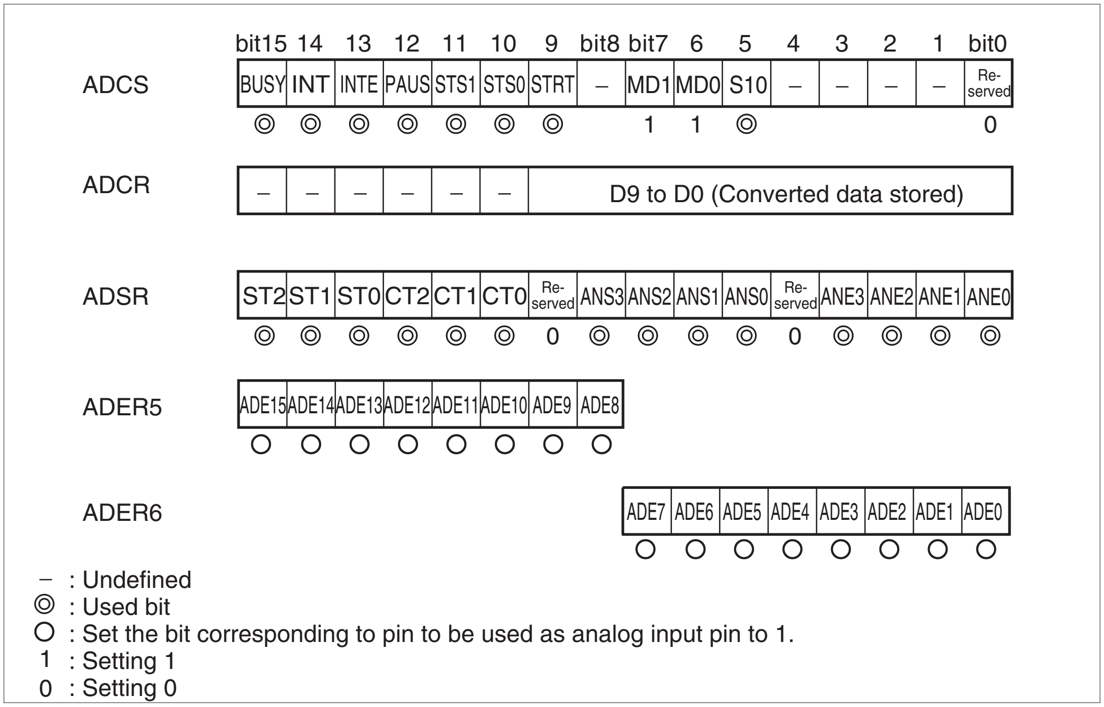
18.5.3 Pause-conversion Mode

In the pause-conversion mode, A/D conversion starts and pauses repeatedly for each channel. When the start trigger is inputted after the A/D conversion pauses at the termination of the A/D conversion for the end channel, A/D conversion is continued after returning to the start channel.

■ Setting of Pause-conversion Mode

Operating the 8-/10-bit A/D converter in the pause-conversion mode requires the setting shown in Figure 18.5-3 .

Figure 18.5-3 Setting of Pause-conversion Mode



	bit15	14	13	12	11	10	9	bit8	bit7	6	5	4	3	2	1	bit0
ADCS	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	–	MD1	MD0	S10	–	–	–	–	Re-served
	◎	◎	◎	◎	◎	◎	◎		1	1	◎					0
ADCR	–	–	–	–	–	–	D9 to D0 (Converted data stored)									
ADSR	ST2	ST1	ST0	CT2	CT1	CT0	Re-served	ANS3	ANS2	ANS1	ANS0	Re-served	ANE3	ANE2	ANE1	ANE0
	◎	◎	◎	◎	◎	◎	0	◎	◎	◎	◎	0	◎	◎	◎	◎
ADER5	ADE15	ADE14	ADE13	ADE12	ADE11	ADE10	ADE9	ADE8								
	○	○	○	○	○	○	○	○								
ADER6									ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0
									○	○	○	○	○	○	○	○

– : Undefined
◎ : Used bit
○ : Set the bit corresponding to pin to be used as analog input pin to 1.
1 : Setting 1
0 : Setting 0

■ Operation of Pause-conversion Mode

- When the start trigger is inputted, A/D conversion starts at the channel set by the A/D conversion start channel select bits (ANS3 to ANS0). The A/D conversion pauses at the termination of the A/D conversion for one channel. When the start trigger is inputted while A/D conversion pauses, A/D conversion for the next channel is performed.
- The A/D conversion pauses at the termination of the A/D conversion for the channel set by the A/D conversion end channel select bits (ANE3 to ANE0). When the start trigger is inputted while A/D conversion pauses, A/D conversion is continued after returning to the channel set by the A/D conversion start channel select bits (ANS3 to ANS0).
- To restart this mode while A/D conversion pauses, input the start trigger set by the A/D start trigger select bits in the A/D control status register (ADCS:STS1, STS0).
- To terminate A/D conversion forcibly, write 0 to the A/D conversion-on flag bit in the A/D control status register (ADCS:BUSY).
- This mode cannot be restarted during A/D conversion.

[When start and end channels are the same]

If the start and end channels have the same channel number (ADCS:ANS3 to ANS0 = ADCS:ANE3 to ANE0), A/D conversion for one channel set as the start channel (= end channel) and pause are repeated.

[Conversion order in pause-conversion mode]

Table 18.5-3 gives an example of the conversion order in the pause-conversion mode.

Table 18.5-3 Conversion Order in Pause-conversion Mode

Start Channel	End Channel	Conversion Order
AN0 pin (ADCS: ANS="0000$_B$")	AN3 pin (ADCS: ANE="0011$_B$")	AN0 → Stop, Start → AN1 → Stop, Start → AN2 → Stop, Start → AN3 → Stop, Start → AN0 → Repeat
AN3 pin (ADCS: ANS="0011$_B$")	AN3 pin (ADCS: ANE="0011$_B$")	AN3 → Stop, Start → AN3 → Stop, Start → Repeat

18.5.4 Conversion Using EI^{2}OS Function

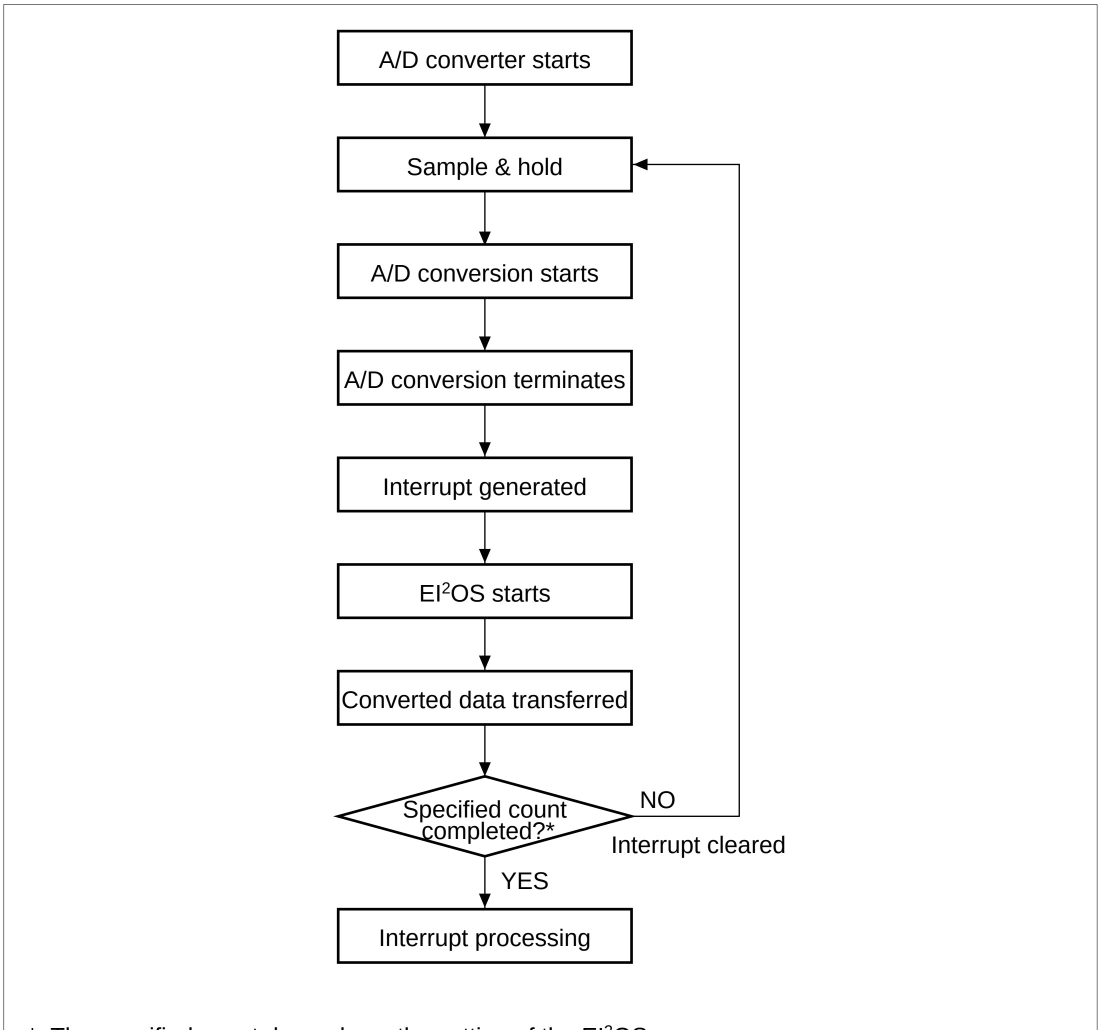
The 8-/10-bit A/D converter can transfer the A/D conversion result to memory by using the EI^{2}OS function.

■ Conversion Using EI^{2}OS

The use of the EI^{2}OS enables the A/D-converted data protection function to transfer multiple data to memory without the loss of converted data even if A/D conversion is performed continuously.

The conversion flow when the EI^{2}OS is used is shown in Figure 18.5-4 .

Figure 18.5-4 Flow of Conversion when Using EI^{2}OS



*: The specified count depends on the setting of the EI^{2}OS.

18.5.5 A/D-converted Data Protection Function

A/D conversion with the output of an interrupt request enabled activates the A/D conversion data protection function.

■ A/D-converted Data Protection Function in 8-/10-bit A/D Converter

The 8-/10-bit A/D converter has only one A/D data register (ADCR) where A/D-converted data is stored. When the A/D conversion results are determined after the termination of A/D conversion, data in the A/D data register is rewritten. Therefore, the A/D conversion results may be lost if the A/D conversion results already stored are not read before data in the A/D data register is rewritten. The A/D-converted data protection function in the 8-/10-bit A/D converter is activated to prevent data loss. This function automatically causes A/D conversion to pause when an interrupt request is generated (ADCS:INT = 1) with an interrupt request enabled (ADCS:INTE = 1).

● A/D-converted data protection function when EI^{2}OS not used

- When the A/D conversion results are stored in the A/D data register (ADCR) after the analog input is A/D-converted, the interrupt request flag bit in the A/D control status register (ADCS: INT) is set to "1".
- The A/D conversion stops for data protection immediately before new data is overwritten to the A/D data register if the interrupt request is enabled (ADCS: INTE = 1) while the interrupt request flag bit set at termination of previous A/D conversion is set at the point that next A/D conversion is terminated.
- When the INT bit is set with an interrupt request from the A/D control status register enabled (ADCS: INTE = 1), an interrupt request is generated. When the INT bit is cleared by the generated interrupt processing, the pause of A/D conversion is cancelled.

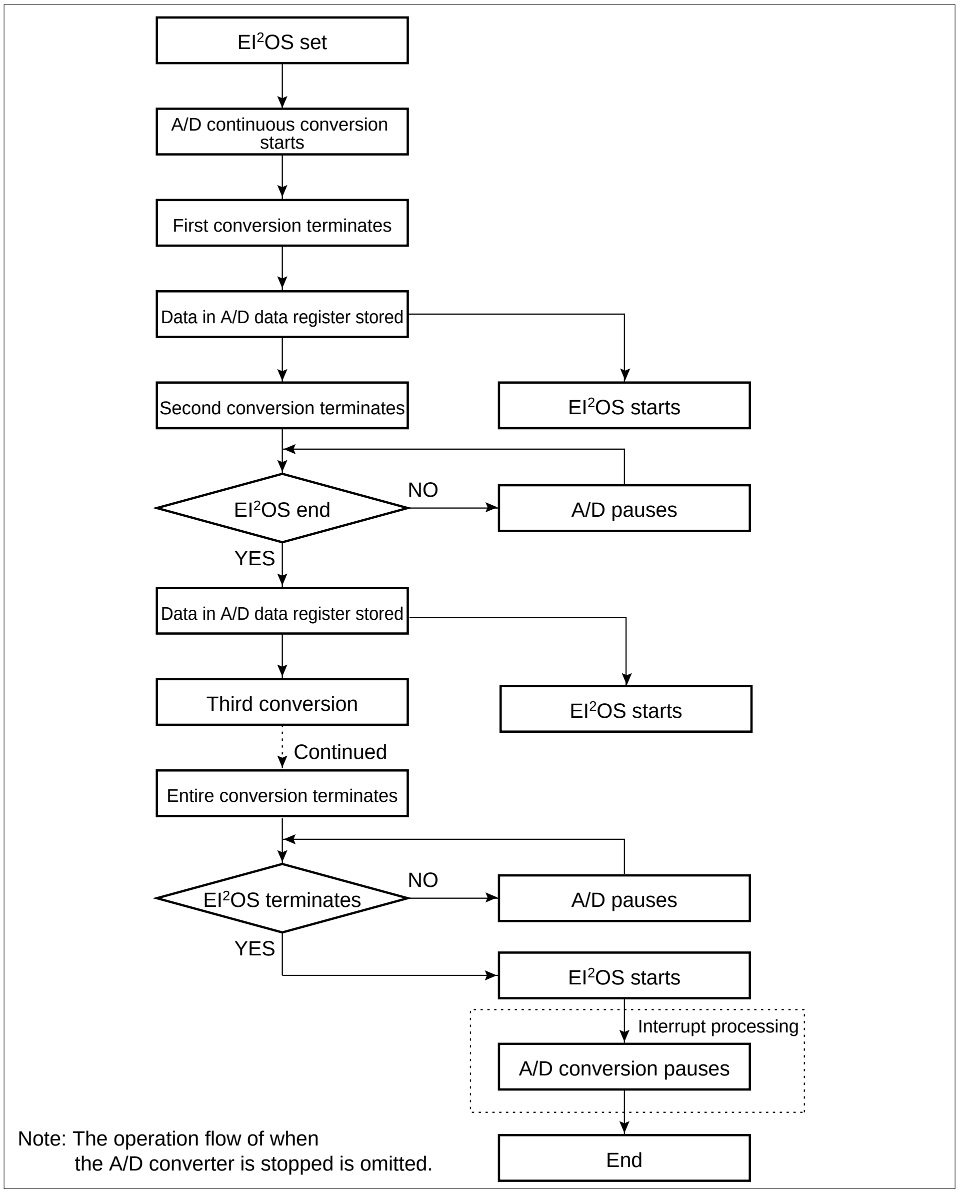
● A/D-converted data protection function when EI^{2}OS used

- The A/D conversion stops for data protection immediately before new data is overwritten to the A/D data register while the EI^{2}OS function is used to transfer the A/D conversion results from the A/D data register to memory when next A/D conversion is terminated. When A/D conversion pauses, the pause flag bit in the A/D control status register (ADCS: PAUS) is set to "1".
- When the transfer of the A/D conversion results to memory by the EI^{2}OS function is terminated, the pause of A/D conversion is cancelled. If the A/D conversion is performed continuously, it is restarted. In this case, the pause flag bit (ADCS:PAUS) is not cleared to "0" automatically. Clearing this bit writes 0 to it.

● Processing flow of A/D conversion data protection function when EI^{2}OS used

Figure 18.5-5 shows the processing flow of the A/D conversion data protection function when the EI^{2}OS is used.

Figure 18.5-5 Processing Flow of A/D Conversion Data Protection Function when Using EI^{2}OS


EI²OS set
A/D continuous conversion starts
First conversion terminates
Data in A/D data register stored
Second conversion terminates
EI²OS starts
EI²OS end
NO
A/D pauses
YES
Data in A/D data register stored
Third conversion
EI²OS starts
Continued
Entire conversion terminates
EI²OS terminates
NO
A/D pauses
YES
EI²OS starts
Interrupt processing
A/D conversion pauses
End
Note: The operation flow of when the A/D converter is stopped is omitted.

Notes:

- The A/D conversion data protection function is activated only when an interrupt request is enabled. Set the interrupt request enable bit in the A/D control status register (ADCS:INTE) to 1.
- The EI^{2}OS function is used to transfer the A/D conversion results to memory, do not disable output of an interrupt request. If output of an interrupt request is disabled during a pause of A/D conversion (ADCS:INTE = 0), A/D conversion may be restarted to rewrite data being transferred.
- The EI^{2}OS function is used to transfer the A/D conversion results to memory, do not restart. Restarting during a pause of A/D conversion may cause loss of the A/D conversion results.

18.6 Precautions when Using 8-/10-bit A/D Converter

Precautions when using the 8-/10-bit A/D converter are given below:

■ Precautions when Using 8-/10-bit A/D Converter

● Analog input pin

- The analog input pins serve as general-purpose I/O ports of port 5 and port 6. When using the pin as an analog input pin, switch the pin to "analog input pin" according to the setting of the analog input enable register (ADER5 , ADER6).
- When using the pin as an analog input pin, write 1 to the bit in the analog input enable register (ADER5, ADER6) corresponding to the pin to be used and set the pin to "analog input enable".
- When an intermediate-level signal is inputted with the pin set as a general-purpose I/O port, the input leakage current flows in the gate. When using the pin as an analog input pin, always set the pin to "analog input enable".

● Precaution when starting by external trigger

- Set the level of the external trigger to inactive ("H" for external trigger) when the A/D start trigger select bits in the A/D control status register (ADCS: STS1 and STS0) is set the same way as starting the 8-/10-bit A/D converter by the external trigger. Holding the input value for the start trigger active may cause the 8-/10-bit A/D converter to start the A/D start trigger select bits in the A/D control status register (ADCS: STS1 and STS0).

● Procedure of 8-/10-bit A/D converter and analog input power-on

- Always apply a power to the 8-/10-bit A/D converter power (AV CC , AVR) and the analog input (AN0 to AN15 pins) after or concurrently with the digital power (V_{CC})-on.
- Always turn off the 8-/10-bit A/D converter power and the analog input before or concurrently with the digital power (V_{CC})-down.
- Note that AVR should not exceed AV_{CC} at power on or power down. (Turning on and off the analog power and digital power simultaneously is enabled.)

● Power supply voltage of 8-/10-bit A/D converter

- To prevent latch up, note that the 8-/10-bit A/D converter power (AV_{CC}) should not exceed the digital power (V_{CC}) voltage.

CHAPTER 19

LOW VOLTAGE DETECTION/ CPU OPERATING DETECTION RESET

This chapter explains the function and operating the low voltage detection/CPU operating detection reset. This function can use only the product with "T" suffix of MB90360 series.

19.1 Overview of Low Voltage/CPU Operating Detection Reset Circuit

The low voltage detection reset circuit watches the power-supply voltage and has the function to detect the power-supply voltages falling lower than the detection voltage values. When the low voltage is detected, internal reset is generated.
When the counter is not cleared within the fixed time after 20-bit counter start that makes the oscillation clock a count clock, CPU operating detection reset circuit generates internal reset.

■ Low Voltage Detection Reset Circuit

Figure 19.1-1 Detection Voltage of Low Voltage/CPU operating Detection Reset Circuit

Detection voltage
4.0V ± 0.3V

After detecting the low voltage, low voltage detection flag (LVRC:LVRF) is set to "1", and internal reset is outputted.

After the low voltage is detected, internal reset is generated and the STOP mode is canceled, because of that keep operating at the STOP mode.

After writing ends, low voltage reset is generated for internal RAM writing period.

■ CPU Operating Detection Reset Circuit

CPU operating detection reset circuit is a counter for preventing the program out of control. After power-on reset, it starts automatically. After it starts, it is necessary to keep clearing regularly within the fixed time. Internal reset is generated when not cleared during the fixed time by an program infinite loop, etc. The width of internal reset generated by CPU operating detection circuit is five machine cycles.

Figure 19.1-2 Interval Time of CPU operating Detection Reset Circuit

Interval time
$2^{20}/F_C$ (approx.262ms)*

*: It is the interval time at oscillation clock 4 MHz.

In the mode that CPU stops operating, the circuit stops.

The counter condition of CPU operating detection reset circuit is indicated as follows.

1) Writing "0" to CL bit of LVRC register

2) Internal reset

3) Oscillation clock stop

4) Transition to sleep mode

5) Transition to timebase timer mode

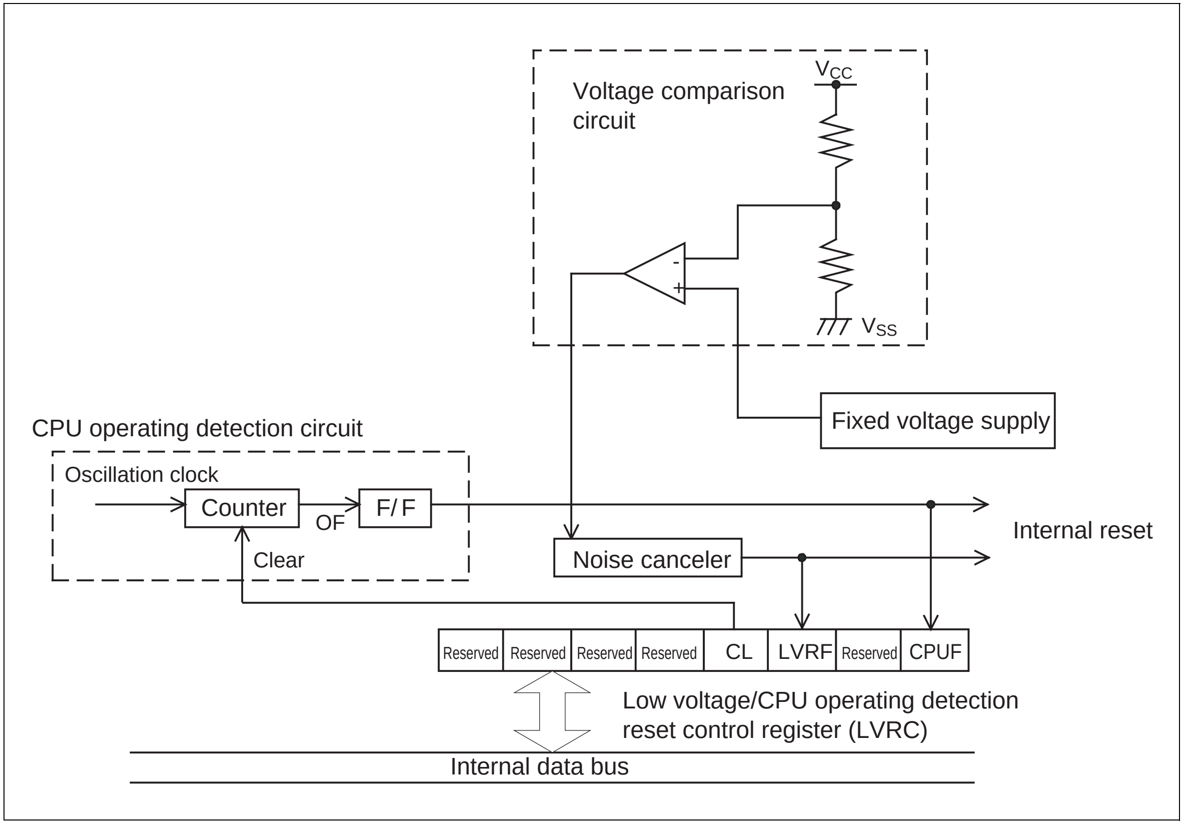
19.2 Configuration of Low Voltage/CPU Operating Detection Reset Circuit

Low voltage/CPU operating detection reset circuit has following three blocks.

- **CPU operating detection circuit**
- **Voltage comparison circuit**
- **Low voltage/CPU operating detection reset control register (LVRC)**

■ Block Diagram of Low Voltage/CPU Operating Detection Reset Circuit

Figure 19.2-1 Block Diagram of Low Voltage/CPU operating Detection Reset Circuit



- CPU operating detection circuit

It is a counter for preventing the program out of control. After it starts, it is necessary to keep clearing regularly within the fixed time.

- Voltage comparison circuit

When the detection voltage is compared with the power-supply voltage, the output is set to "H" after the low voltage detection.

After the power supply is turned on, it always operates.

- Low voltage/CPU operating detection reset control register (LVRC)

This register clears the low voltage/CPU operating detection reset flag and the counter of CPU operating detection function.

- Reset factor of low voltage/CPU operating detection reset circuit

After power-supply voltages is fallen more than the detection voltages, internal reset is generated.

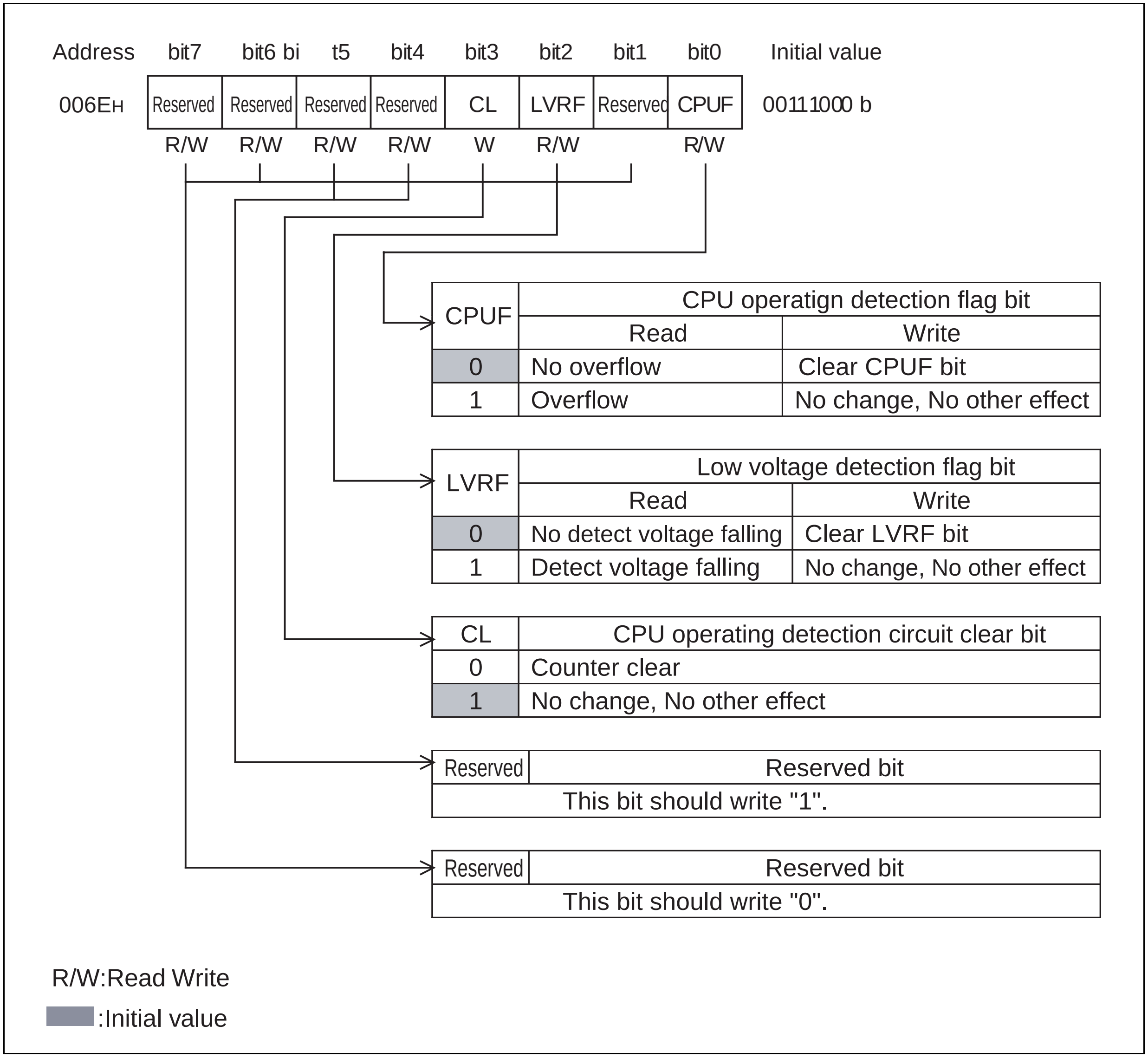
When the counter of CPU operating detection circuit is not cleared during the fixed time, internal reset is generated.

19.3 Low Voltage/CPU Operating Detection Reset Circuit Register

This register clears the low voltage/CPU operating detection reset flag and the counter of CPU operating detection circuit.

■ Low Voltage/CPU Operating Detection Reset Control Register (LVRC)

Figure 19.3-1 **Low Voltage/CPU operating Detection Reset Control Register (LVRC)**



Address	bit7	bit6 bi	t5	bit4	bit3	bit2	bit1	bit0	Initial value
006EH	Reserved	Reserved	Reserved	Reserved	CL	LVRF	Reserved	CPUF	00111000 b
	R/W	R/W	R/W	R/W	W	R/W		R/W	

CPUF	CPU operatign detection flag bit	
	Read	Write
0	No overflow	Clear CPUF bit
1	Overflow	No change, No other effect

LVRF	Low voltage detection flag bit	
	Read	Write
0	No detect voltage falling	Clear LVRF bit
1	Detect voltage falling	No change, No other effect

CL	CPU operating detection circuit clear bit
0	Counter clear
1	No change, No other effect

Reserved	Reserved bit
This bit should write "1".	

Reserved	Reserved bit
This bit should write "0".	

R/W:Read Write

▨ :Initial value

Table 19.3-1 Functional Description of Low Voltage/CPU operating Detection Reset Control Register

Bit name		Function
bit7/ bit6	Reserved: Reserved bits	Note: These bits should write "0".
bit5/ bit4	Reserved: Reserved bits	Note: These bits should write "1".
bit3	CL: CPU operating detection clear bit	This bit is a bit that clears the counter of CPU operating detection circuit. When "0" is written in the CL bit, the counter of CPU operating detection circuit is cleared.
bit2	LVRF: Low voltage detection flag bit	When falling of the power-supply voltage is detected, the LVRF bit is set to "1". This bit is cleared by "0" at write. And even if "1" is written in this bit, the LVRF bit is no effect. This bit is not initialized in internal reset, and it is initialized only by the external reset input.
bit1	Reserved: Reserved bit	Note: This bit should write "0".
bit0	CPUF: CPU	When the counter of CPU operating detecting function overflows, the CPUF bit is set to "1". This bit is cleared by "0" at write. And even if "1" is written in this bit, the CPUF bit is no effect. This bit is not initialized in internal reset, and it is initialized only by the external reset input.

19.4 Operating of Low Voltage/CPU Operating Detection Reset Circuit

The circuit watches the power-supply voltage. When the power supply voltage is lower than the set value, internal reset is generated. In CPU operating detecting function, internal reset is generated without the counter clear at constant intervals. When internal reset is generated by the detection of the low voltage or the out of CPU control, the content of the register is not guaranteed. The program restarts from the address specified by the reset vector after the reset sequence is executed when the low voltage reset is canceled.

■ Operating of Low Voltage/CPU Operating Detection Reset Circuit

The low voltage detection reset circuit starts the detection of the low voltage without taking the operating stability wait time after reset is canceled.

■ Operating of CPU Operating Detection Reset Circuit

The CPU operating detection reset circuit starts the detection of the CPU operation without taking the operating stability wait time after reset is canceled.

Note:

The current is consumed during the sleep or stop mode because the low voltage reset circuit always operates.

19.5 Notes on Using Low Voltage/CPU Operating Detection Reset Circuit

This section explains the note on using the low voltage/CPU operating detection reset circuit.

■ Notes on Using Low Voltage Detection Reset Circuit

● Disabled operating stop from program

The low voltage detection reset circuit operates continuously after the power supply is turned on and the operating stabilization wait time passes. Operating can not be stopped with software.

● Operating at STOP mode

Low voltage detection reset keeps operating at the STOP mode. Therefore, if a low voltage is detected in the STOP mode, reset is generated and the STOP mode is canceled.

■ Notes on Using CPU Operating Detection Reset Circuit

● Disabled operating stop from program

CPU operating detection reset circuit operates continuously after turning on the power supply. Operating cannot be stopped with software.

● Reset generation control of CPU operating detecting function

CPU operating detecting function should clear the counter at regular intervals. The counter is cleared by writing "0" to the CL bit of the LVRC register, and the reset generation can be controlled.

● Stop and clear of counter

In the mode CPU stops operating, CPU operating detecting function clears the counter and stops operating.

● Operation in sub-oscillation mode

The CPU operation detection function will stop operation in sub-oscillation mode. For this reason, also use the watchdog reset function.

19.6 Sample Program for Low Voltage/CPU Operating Detection Reset Circuit

This section shows the sample program for low voltage/CPU operating detection reset circuit.

■ Sample Program for Low Voltage/CPU Operating Detection Reset Circuit

● Processing specification

The counter of CPU operating detecting function is cleared.

● Coding example

```
LVRC    EQU    006EH    ; Address of low voltage/CPU operating detection reset control register
------------------------------------Main program------------------------------------
               CSEG     ; [CODE SEGMENT]
                 :
        MOV    LVRC,#00110101B
                 :
        END
```

CHAPTER 20

LIN-UART

This chapter explains the functions and operation of LIN-UART.

20.1 Overview of LIN-UART

The LIN-UART with LIN (Local Interconnect Network) - Function is a general-purpose serial data communication interface for performing synchronous or asynchronous communication (start-stop synchronization) with external devices. LIN-UART provides bidirectional communication function (normal mode), master-slave communication function (multiprocessor mode in master/slave systems), and special features for LIN-bus systems.

■ LIN-UART Functions

● LIN-UART functions

LIN-UART is a general-purpose serial data communication interface for transmitting serial data to and receiving data from another CPU and peripheral devices. It has the functions listed in Table 20.1-1 .

Table 20.1-1 LIN-UART Functions (1/2)

	Function
Data buffer	Full-duplicate double-buffer
Serial input	Perform oversampling 5 times and determine the received value by majority decision of sampling time (asynchronous mode only)
Transfer mode	• Synchronous to clock (selecting start/stop synchronous or start/stop bit) • Asynchronous (start/stop bits can be used.)
Baud rate	• Dedicated baud-rate generator (The baud rate is consisted of 15-bit reload counter.) • An external clock can be inputted and also be adjusted by reload counter.
Data length	• 7 bits (other than synchronous or LIN mode) • 8 bits
Signal type	NRZ (Non Return to Zero)
Start bit timing	Synchronization to the falling edge of the start bit in the asynchronous mode
Detection of receive error	• Framing error • Overrun error • Parity error (not supported for operation mode 1)
Interrupt request	• Receive interrupt (receive termination, detection of receive error, LIN Synch break detection) • Transmit interrupt (transmit data empty) • Interrupt request to ICU (LIN Synch field detection: LSYN) • Both the transmission and reception support EI^2OS
Master/slave type communication function (multiprocessor mode)	This function enables communication between 1 (only use master) and n (slave) (This function supports for the both of master and slave system.)
Synchronous mode	Master of slave function
Pin access	Capable of reading the state of serial I/O pin directly

Table 20.1-1 LIN-UART Functions (2/2)

	Function
LIN bus option	• Master device operation • Slave device operation • LIN Synch break detection • LIN Synch break generation • Detection of start/stop edges in LIN Synch field connected to input capture 0 and 1
Synchronous serial clock	Synchronous serial clock can be continuously outputted to SCK pin for synchronous communication with start/stop bits.
Clock delay option	Special synchronous clock mode for delaying clock (useful to SPI)

■ LIN-UART operation modes

The LIN-UART operates in four different modes, which are determined by the MD0- and the MD1-bit of the serial mode register (SMR). Mode 0 and 2 are used for bidirectional serial communication, mode 1 for master/slave communication and mode 3 for LIN master/slave communication.

Table 20.1-2 Operation Mode of LIN-UART

Operation Mode		Data Length		Synchronous/ Asynchronous	Length of Stop Bit	Data Bit Format
		No Parity	With Parity			
0	Normal mode	7 or 8 bits		Asynchronous	1 bit or 2 bits	LSB first MSB first
1	Multiprocessor mode	7 or 8 bits-+1 *	-	Asynchronous		
2	Normal mode	8		Synchronous	None, 1 bit, 2 bits	
3	LIN mode	8	-	Asynchronous	1 bit	LSB first

- : Setting disabled
*: +1 is the address/data select bit (A/D) used for controlling communication in multiprocessor mode.

The MD1 and MD0 bits of the serial mode register (SMR) determine the operation mode of LIN-UART as shown in the following table:

Table 20.1-3 Operation Mode of LIN-UART

MD1	MD0	Mode	Type
0	0	0	Asynchronous (normal mode)
0	1	1	Asynchronous (multiprocessor mode)
1	0	2	Synchronous (normal mode)
1	1	3	Asynchronous (LIN mode)

Note:

Mode 1 operation is supported both for master or slave operation of LIN-UART in a master-slave connection system. In Mode 3 the LIN-UART function is locked to 8N1-Format, LSB first.

If the mode is changed, LIN-UART cuts off all possible transmission or reception and awaits then new action.

■ LIN-UART interrupt and EI^{2}OS

Table 20.1-4 LIN-UART Interrupt and EI^{2}OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI^{2}OS
		Register name	Address	Lower	Upper	Bank	
LIN-UART0 reception	#35(23$_H$)	ICR12	0000BC$_H$	FFFF70$_H$	FFFF71$_H$	FFFF72$_H$	*1
LIN-UART0 transmission	#36(24$_H$)	ICR12	0000BC$_H$	FFFF6C$_H$	FFFF6D$_H$	FFFF6E$_H$	*2
LIN-UART1 reception	#37(25$_H$)	ICR13	0000BD$_H$	FFFF68$_H$	FFFF69$_H$	FFFF6A$_H$	*1
LIN-UART1 transmission	#38(26$_H$)	ICR13	0000BD$_H$	FFFF64$_H$	FFFF65$_H$	FFFF66$_H$	*2

*1: EI^{2}OS service is usable if the other interrupt (s) which shares the ICR12 to ICRB and same interrupt vector is (are) not enabled. Detection of receive errors is possible and stop function for EI^{2}OS service is supported.

*2: EI^{2}OS service is usable if the other interrupt (s) which shares the ICR12 to ICRB and same interrupt vector is (are) not enabled.

20.2 Configuration of LIN-UART

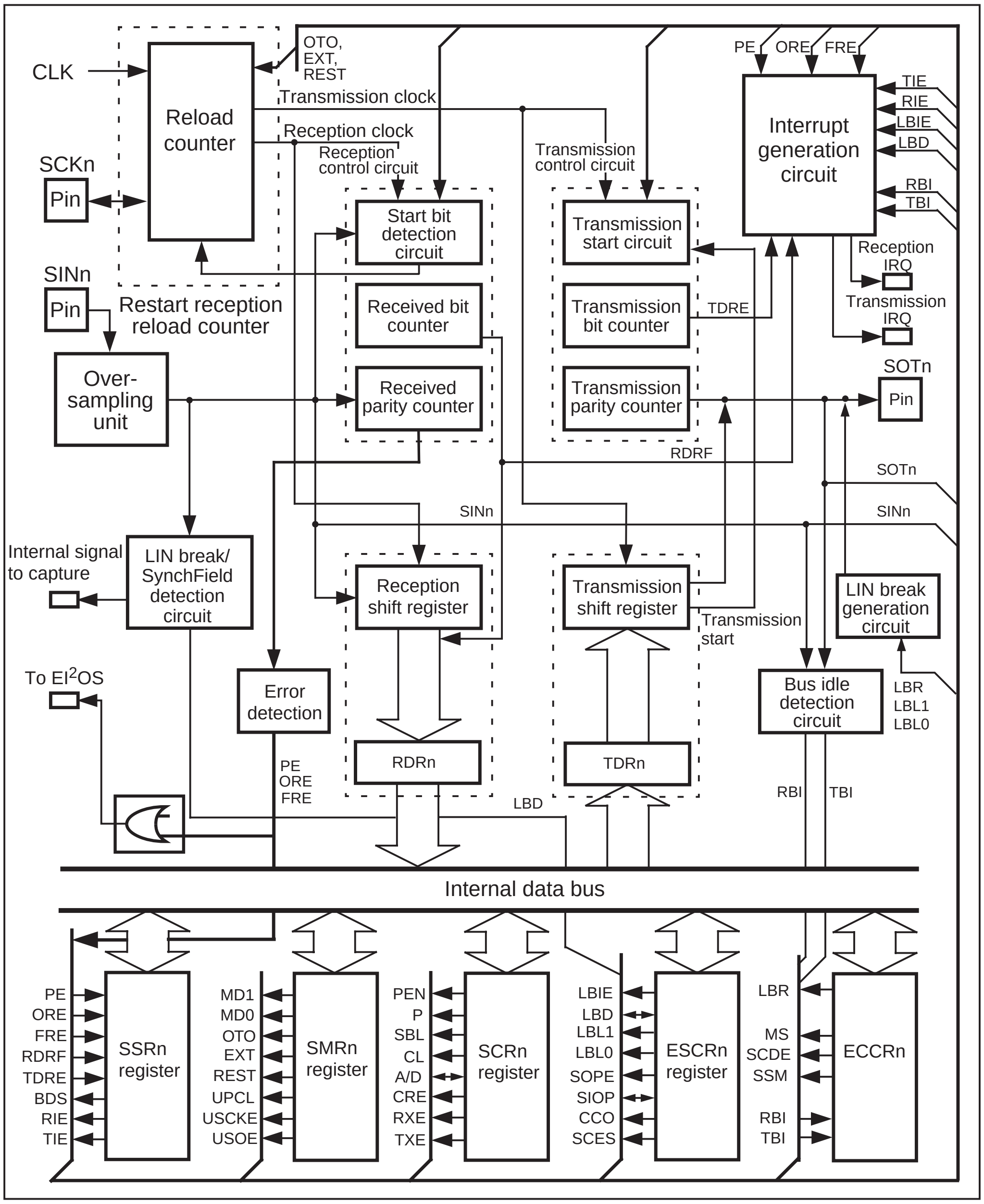
This section provides a short overview on the building blocks of LIN-UART.

LIN-UART consists of the following blocks:

- Reload Counter
- Reception Control Circuit
- Reception Shift Register
- Reception Data Register (RDR)
- Transmission Control Circuit
- Transmission Shift Register
- Transmission Data Register (TDR)
- Error Detection Circuit
- Oversampling circuit
- Interrupt Generation Circuit
- LIN Synch Break/Synch Field Detection Circuit
- LIN Synch Break Generation Circuit
- Bus Idle Detection Circuit
- LIN-UART Serial Mode Register (SMR)
- Serial Control Register (SCR)
- Serial Status Register (SSR)
- Extended Com. Contr. Reg. (ECCR)
- Extended Status/Contr. Reg. (ESCR)

■ Block Diagram of LIN-UART

Figure 20.2-1 Block Diagram of LIN-UART


CLK
SCKn
Pin
SINn
Pin
Reload counter
OTO, EXT, REST
Transmission clock
Reception clock
Restart reception reload counter
Over-sampling unit
Reception control circuit
Start bit detection circuit
Received bit counter
Received parity counter
Transmission control circuit
Transmission start circuit
Transmission bit counter
Transmission parity counter
TDRE
RDRF
PE
ORE
FRE
Interrupt generation circuit
TIE
RIE
LBIE
LBD
RBI
TBI
Reception IRQ
Transmission IRQ
SOTn
Pin
SOTn
SINn
SINn
Internal signal to capture
LIN break/ SynchField detection circuit
To EI2OS
Error detection
PE ORE FRE
Reception shift register
RDRn
Transmission shift register
Transmission start
TDRn
LBD
LIN break generation circuit
LBR LBL1 LBL0
Bus idle detection circuit
RBI
TBI
Internal data bus
PE ORE FRE RDRF TDRE BDS RIE TIE
SSRn register
MD1 MD0 OTO EXT REST UPCL USCKE USOE
SMRn register
PEN P SBL CL A/D CRE RXE TXE
SCRn register
LBIE LBD LBL1 LBL0 SOPE SIOP CCO SCES
ESCRn register
LBR MS SCDE SSM RBI TBI
ECCRn

n = 0, 1

■ Explanation of the different blocks

● Reload Counter

The reload counter is a 15-bit reload counter that functions as the dedicated baud rate generator. It can select external clock or internal clock for the transmitting and receiving clocks. The reload counter has a 15-bit register for the reload value. The actual count of the transmission reload counter can be read via the BGRn0/n1.

● Reception Control Circuit

The reception control circuit consists of a received bit counter, start bit detection circuit, and received parity counter. The received bit counter counts reception data bits. When reception of one data item for the specified data length is completed, the received bit counter sets the reception data register full flag. In this case, if the reception interrupt is enabled, the reception interrupt request is generated. The start bit detection circuit detects start bits from the serial input signal and sends a signal to the reload counter to synchronize it to the falling edge of these start bits. The received parity counter calculates the parity of the reception data.

● Reception Shift Register

The reception shift register fetches reception data input from the SINn pin, shifting the data bit by bit. When reception is completed, the reception shift register transfers receive data to the RDR register.

● Reception Data Register (RDR)

This register retains reception data. Serial input data is converted and stored in this register.

● Transmission Control Circuit

The transmission control circuit consists of a transmission bit counter, transmission start circuit, and transmission parity counter. The transmission bit counter counts transmission data bits. When the transmission of one data item of the specified data length is completed, the transmission bit counter sets the transmission data register full flag. In this case, if the transmission interrupt is enabled, the transmission interrupt request is generated. The transmission start circuit starts transmission when data is written to TDR register. The transmission parity counter generates a parity bit for data to be transmitted if parity is enabled.

● Transmission Shift Register

The transmission shift register transfers data written to the TDR register to itself and outputs the data to the SOTn pin, shifting the data bit by bit.

● Transmission Data Register (TDR)

This register sets transmission data. Data written to this register is converted to serial data and outputted.

● Error Detection Circuit

The error detection circuit checks if there was any error during the last reception. If an error has occurred it sets the corresponding error flags.

● Oversampling Circuit

The oversampling circuit oversamples the incoming data at the SINn pin for five times in the asynchronous mode. The received value is determined by majority decision of sampling time. It is switched off in synchronous operation mode.

● Interrupt Generation Circuit

The interrupt generation circuit administers all cases of generating a reception or transmission interrupt. If a corresponding interrupt enable bit is set, the interrupt will be generated immediately.

● LIN synch Break and Synchronization Field Detection Circuit

The LIN break and LIN synchronization field detection circuit detects a LIN synch break if a LIN master node is sending a message header. If a LIN synch break is detected LBD flag bit is generated. The first and the fifth falling edge of the LIN synchronization field is recognized by this circuit by generating an internal signal for the Input Capture Unit to measure the actual serial clock synchronization of the transmitting master node.

● LIN Synch Break Generation Circuit

The LIN break generation circuit generates a LIN synch break of a determined length.

● Bus Idle Detection circuit

The bus idle detection circuit recognizes if neither reception nor transmission is going on. In this case, the circuit generates the special flag bits TBI and RBI.

● LIN-UART Serial Mode Register (SMR)

This register performs the following operations:

- Selecting the LIN-UART operation mode
- Selecting a clock input source
- Selecting if an external clock is connected "one-to-one" or connected to the reload counter
- Resetting dedicated reload timer
- Resetting the LIN-UART software (preserving the settings of the registers)
- Specifying whether to enable serial data output to the corresponding pin
- Specifying whether to enable clock output to the corresponding pin

● Serial Control Register (SCR)

This register performs the following operations:

- Specifying whether to provide parity bits
- Selecting parity bits
- Specifying a stop bit length
- Specifying a data length
- Selecting a frame data format in mode 1
- Clearing the error flags
- Specifying whether to enable transmission
- Specifying whether to enable reception

● Serial Status Register (SSR)

This register performs the following functions;

- Indicating status of receive/transmit operations and errors
- Specifying LSB first or MSB first
- Receive interrupt enable/disable
- Transmit interrupt enable/disable

● Extended Communication Control Register (ECCR)

This register performs the following functions;

- Indicating bus idle detection
- Specifying synchronous clock
- Specifying LIN synch break generation

● Extended Status/Control Register (ESCR)

This register performs the following functions;

- LIN synch break interrupt enable/disable
- Indicating LIN synch break detection
- Specifying LIN synch break length
- Directly accessing SINn and SOTn pins
- Specifying continuous clock output operation in LIN-UART synchronous clock mode
- Specifying sampling clock edge

20.3 LIN-UART Pins

This section describes the LIN-UART pins and provides a pin block diagram.

■ LIN-UART Pins

The LIN-UART pins also serve as general ports. Table 20.3-1 lists the pin functions, I/O formats, and settings required to use LIN-UART.

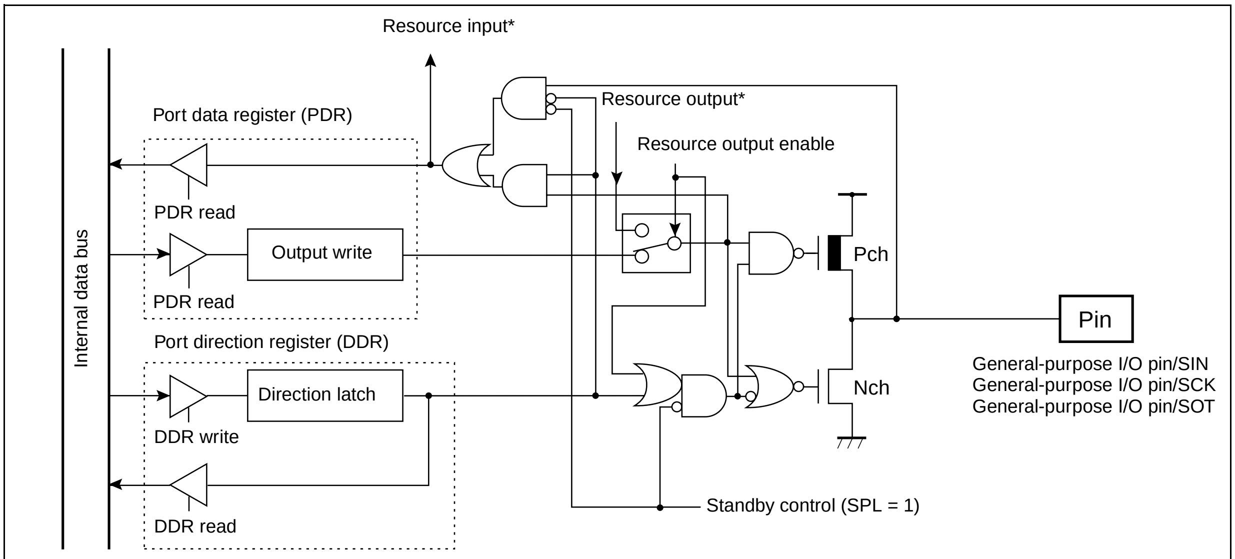
Table 20.3-1 LIN-UART Pin

Pin Name	Pin Function	I/O Format	Standby Control	Setting Required to Use Pin
P82/SIN0 P85/SIN1	Port I/O or serial data input	CMOS output/CMOS, Automotive input	Provided	Set as input port (DDR: corresponding bit = 0)
P83/SOT0 P86/SOT1	Port I/O or serial data output			Set to output enable mode (SMRn: SOE = 1)
P84/SCK0 P87/SCK1	Port I/O or serial clock input/output			Set as an input port when a clock is inputted (DDR: corresponding bit = 0)
				Set to output enable mode when a clock is outputted (SMRn: SCKE = 1)

See "3. DC Characteristics in ELECTRICAL CHARACTERISTICS" in the data sheet for the standard value.

■ Block Diagram of LIN-UART Pins

Figure 20.3-1 Block Diagram of LIN-UART Pins



Standby control: Stop mode (SPL=1), watch mode (SPL=1), timebase timer mode (SPL=1)

*: Resource I/O signals are inputted or outputted from pins having peripheral functions.

20.4 LIN-UART Registers

The following figure shows the LIN-UART registers.

■ LIN-UART Registers

Figure 20.4-1 LIN-UART Registers

- LIN-UART0

Address:		bit 15 bit 8	bit 7 bit 0
000021_H	000020_H	SCR0 (serial control register)	SMR0 (serial mode register)
000023_H	000022_H	SSR0 (serial status register)	RDR0/TDR0 (reception data register/transmission data register)
000025_H	000024_H	ESCR0 (Extended status control register)	ECCR0 (Extended communication control register)
000027_H	000026_H	BGR01 (Baud rate generator register)	BGR00 (Baud rate generator register)

- LIN-UART1

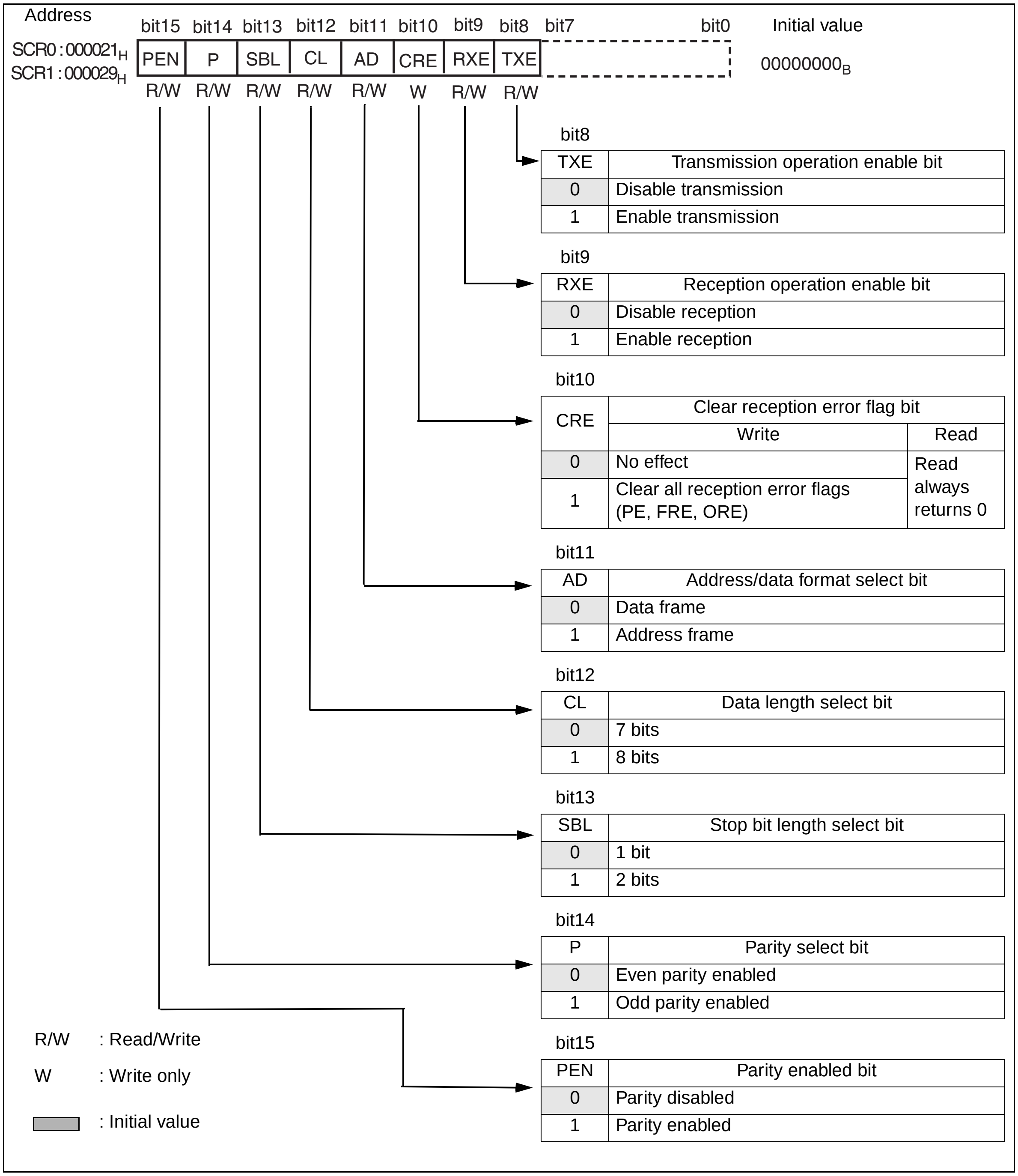
Address:		bit 15 bit 8	bit 7 bit 0
000029_H	000028_H	SCR1 (serial control register)	SMR1 (serial mode register)
$00002B_H$	$00002A_H$	SSR1 (serial status register)	RDR1/TDR1 (reception data register/transmission data register)
$00002D_H$	$00002C_H$	ESCR1 (Extended status control register)	ECCR1 (Extended communication control register)
$00002F_H$	$00002E_H$	BGR11(Baud rate generator register)	BGR10(Baud rate generator register)

20.4.1 Serial Control Register (SCR)

This register specifies parity bits, selects the stop bit and data lengths, selects a frame data format in mode 1, clears the reception error flag, and specifies whether to enable transmission and reception.

■ Serial Control Register (SCR)

Figure 20.4-2 Configuration of the Serial Control Register (SCR)


Address
SCR0 : 000021H
SCR1 : 000029H
bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8 bit7 bit0
PEN P SBL CL AD CRE RXE TXE
R/W R/W R/W R/W R/W W R/W R/W
Initial value
00000000B

bit8

TXE	Transmission operation enable bit
0	Disable transmission
1	Enable transmission

bit9

RXE	Reception operation enable bit
0	Disable reception
1	Enable reception

bit10

CRE	Clear reception error flag bit	
	Write	Read
0	No effect	Read always returns 0
1	Clear all reception error flags (PE, FRE, ORE)	

bit11

AD	Address/data format select bit
0	Data frame
1	Address frame

bit12

CL	Data length select bit
0	7 bits
1	8 bits

bit13

SBL	Stop bit length select bit
0	1 bit
1	2 bits

bit14

P	Parity select bit
0	Even parity enabled
1	Odd parity enabled

bit15

PEN	Parity enabled bit
0	Parity disabled
1	Parity enabled

R/W : Read/Write

W : Write only

▨ : Initial value

Table 20.4-1 Function of Each Bit in Serial Control Register (SCR)

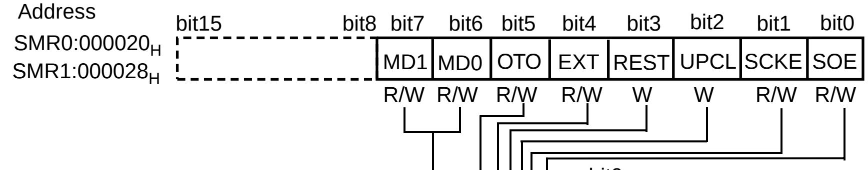
No.	Bit Name	Function
bit15	PEN: Parity enable bit	This bit selects whether to add a parity bit during transmission or detect it during reception. Note: Parity bit is only provided in mode 0 and in mode 2 if SSM of the ECCR is selected to 1. This bit is fixed to 0 (no parity) in mode 3 (LIN).
bit14	P: Parity selection bit	When parity is provided, this bit selects even (0) or odd (1) parity
bit13	SBL: Stop bit length selection bit	This bit selects the length of the stop bit of an asynchronous data frame or a synchronous frame if SSM of the ECCR is selected 1. This bit is fixed to 0 (1 stop bit) in mode 3 (LIN). Note: At reception, first stop bit is always detected.
bit12	CL: Data length selection bit	This bit specifies the length of transmission or reception data. This bit is fixed to 1 (8 bits) in mode 2 and 3.
bit11	AD: Address/Data format selection bit	This bit specifies the frame data format to be transmitted and received in multiprocessor mode 1. Writing to this bit is provided for a master CPU, reading from it for slave CPU. A 1 indicates an address data frame, a 0 indicates a usual data frame. The reading value is a value of last received data format. Note: Please read the hints about using this bit in "20.8 Notes on Using LIN-UART".
bit10	CRE: Clear reception error flag bit	This bit clears the FRE, ORE, and PE flags of the Serial Status Register (SSR). Writing a 1 to it clears the error flag. Writing a 0 has no effect. Reading from it always returns 0. Note; Clear reception error flags after the receive operation.
bit9	RXE: Reception operation enable bit	This bit enables/disables LIN-UART reception. If this bit is set to 0, LIN-UART disables the reception of data frames. If this bit is set to 1, LIN-UART enables the reception of data frames. The LIN synch break detection in mode 3 remains unaffected. Note: If reception is disabled (RXE=0) during receiving, it is stopped immediately. In this case, data is not guaranteed.
bit8	TXE: Transmission operation enable bit	This bit enables/disables LIN-UART transmission. If the bit is set to 0, LIN-UART disables the transmission of data frames. If the bit is set to 1, LIN-UART enables the transmission of data frames. Note: If transmission is disabled (TXE=0) during transmitting, it is stopped immediately. In this case, data is not guaranteed.

20.4.2 LIN-UART Serial Mode Register (SMR)

This register selects an operation mode and baud rate clock and specifies whether to enable output of serial data and clocks to the corresponding pin.

■ LIN-UART Serial Mode Register (SMR)

Figure 20.4-3 Configuration of the Serial Mode Register (SMR)



bit0

SOE	Serial data output enable bit of LIN-UART
0	General-purpose I/O port
1	Serial data output enable pin of LIN-UART

bit1

SCKE	LIN-UART serial clock output enable bit
0	General-purpose I/O port or LIN-UART clock input pin
1	Serial clock output pin of LIN-UART

bit2

UPCL	LIN-UART programmable clear bit	
	write	read
0	Ignored	always read 0
1	Reset LIN-UART	

bit3

REST	Restart dedicated Reload Counter bit	
	write	read
0	Ignored	always read 0
1	Restart Counter	

bit4

EXT	External Serial Clock Source select bit
0	Use internal Baud Rate Generator (Reload Counter)
1	Use external Serial Clock Source

bit5

OTO	One-to-one external clock Input enable bit
0	Use external Clock with Baud Rate Generator (Reload Counter)
1	Use external Clock as it is

bit7	bit6	
MD1	MD0	Operation Mode Setting bit
0	0	Mode 0: Asynchronous normal
0	1	Mode 1: Asynchronous Multiprocessor
1	0	Mode 2: Synchronous
1	1	Mode 3: Asynchronous LIN

R/W : Read/Write

W : Write only

▨ : Initial value

Table 20.4-2 Function of Each Bit in Serial Mode Register (SMR)

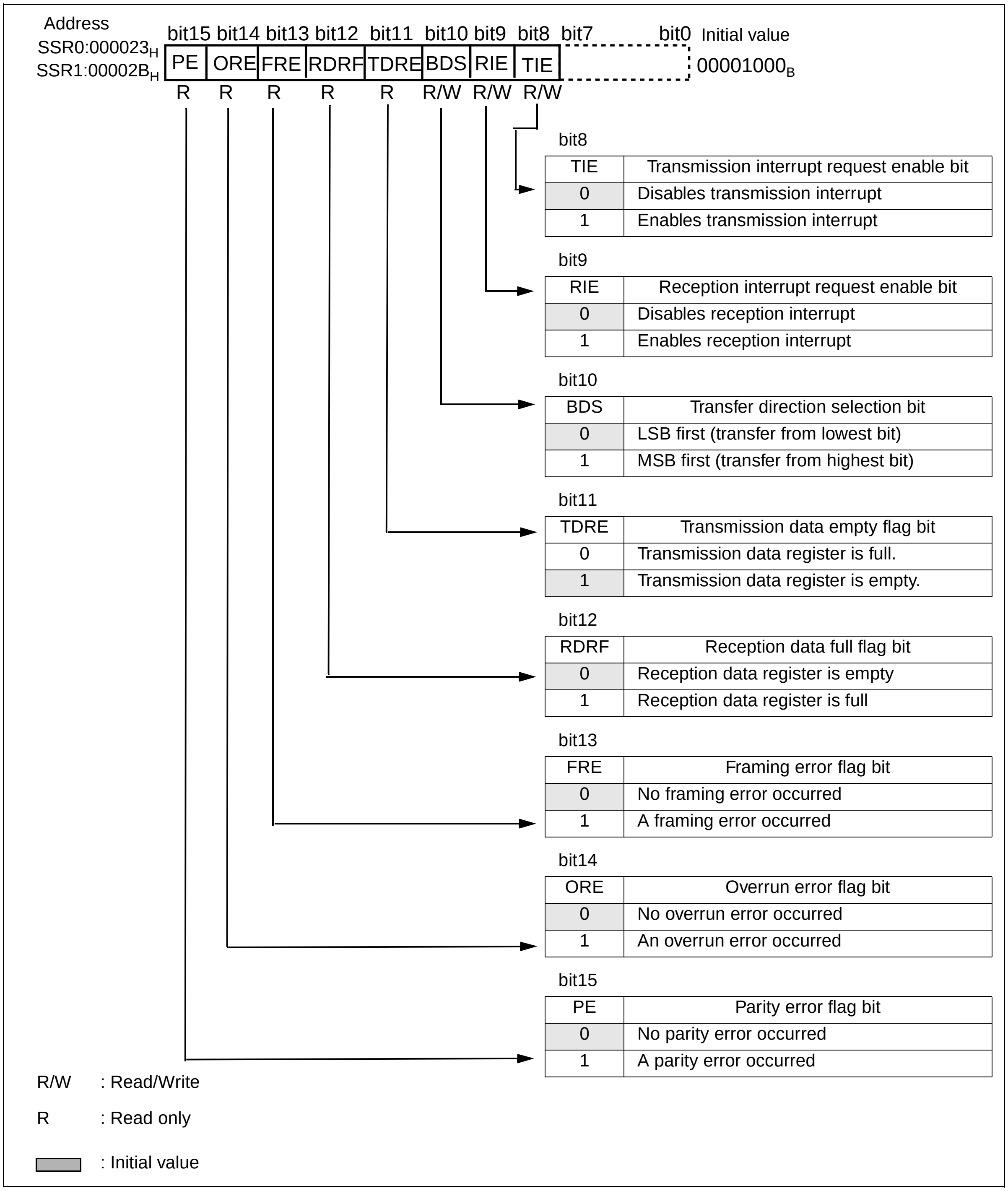
No.	Bit name	Function
bit7, bit6	MD1, MD0: Operation mode setting bits	These two bits set the LIN-UART operation mode.
bit5	OTO: One-to-one external clock input enable bit	This bit sets an external clock directly to the LIN-UART serial clock by writing "1". This function is used for operating mode 2 (synchronous) slave mode operation (ECCR:MS=1). When EXT=0, this bit is fixed to "0".
bit4	EXT: External serial clock source selection bit	This bit selects the clock input. When "0" is set to this bit, it selects the clock of internal baud rate generator (reload counter). When "1" is set to it, it selects the external serial clock source.
bit3	REST: Restart of dedicated reload counter bit	If a 1 is written to this bit, the reload counter is restarted. Writing 0 to it has no effect. Reading from this bit always returns 0.
bit2	UPCL : LIN-UART programmable clear bit (LIN-UART software reset)	Writing a 1 to this bit resets LIN-UART immediately (LIN-UART software reset). The register settings are preserved. Possible reception or transmission will cut off. All flags (TDRE, RDRF, LBD, PE, ORE, FRE) are cleared. Reset the LIN-UART after disabling the interrupt and transmission. Also, when the reception data register is cleared (RDR = 00_H), the reload counter restarts. Writing 0 to this bit has no effect. Reading from it always returns 0.
bit1	SCKE: LIN-UART serial clock output enable bit	This bit controls the serial clock I/O ports. When this bit is 0, SCKn pin operates as general-purpose I/O port or serial clock input pin. When this bit is 1, the pin operates as serial clock output pin and outputs clock in operating mode 2 (synchronous). Note: When using SCKn pin as serial clock input (SCKE=0) pin, set the corresponding DDR bit of general-purpose port as input port. Also, select external clock (EXT = 1) using the external clock selection bit. Reference: When the SCKn pin is assigned to serial clock output (SCKE=1), it functions as the serial clock output pin regardless of the status of the general-purpose I/O ports.
bit0	SOE: LIN-UART serial data output enable bit	This bit enables or disables the output of serial data. When this bit is 0, SOTn pin operates as general-purpose I/O port. When this bit is 1, SOTn pin operates as serial data output pins (SOTn). Reference: When the output of serial data is enabled (SOE=1), SOTn pin functions as serial data output pin (SOTn) regardless of the status of general-purpose I/O ports.

20.4.3 Serial Status Register (SSR)

This register checks the transmission and reception status and error status, and enables and disables the transmission and reception interrupts.

■ Serial Status Register (SSR)

Figure 20.4-4 Configuration of the Serial Status Register (SSR)


Address
SSR0:000023_H
SSR1:$00002B_H$
bit15 bit14 bit13 bit12 bit11 bit10 bit9 bit8 bit7 bit0
PE ORE FRE RDRF TDRE BDS RIE TIE
R R R R R R/W R/W R/W
Initial value
00001000_B

bit8

TIE	Transmission interrupt request enable bit
0	Disables transmission interrupt
1	Enables transmission interrupt

bit9

RIE	Reception interrupt request enable bit
0	Disables reception interrupt
1	Enables reception interrupt

bit10

BDS	Transfer direction selection bit
0	LSB first (transfer from lowest bit)
1	MSB first (transfer from highest bit)

bit11

TDRE	Transmission data empty flag bit
0	Transmission data register is full.
1	Transmission data register is empty.

bit12

RDRF	Reception data full flag bit
0	Reception data register is empty
1	Reception data register is full

bit13

FRE	Framing error flag bit
0	No framing error occurred
1	A framing error occurred

bit14

ORE	Overrun error flag bit
0	No overrun error occurred
1	An overrun error occurred

bit15

PE	Parity error flag bit
0	No parity error occurred
1	A parity error occurred

R/W : Read/Write

R : Read only

▨ : Initial value

Table 20.4-3 Function of Each Bit in Serial Status Register (SSR)

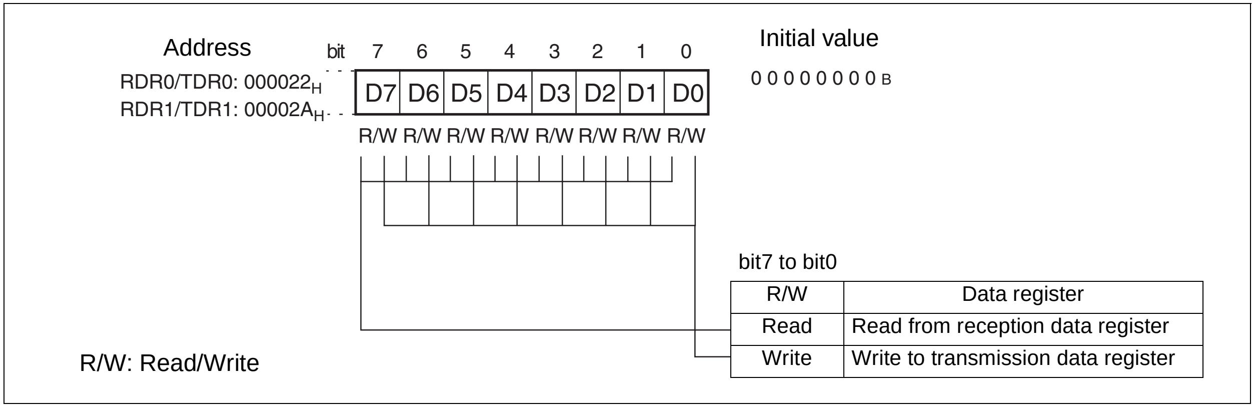
No.	Bit name	Function
bit15	PE: Parity error flag bit	• This bit is set to 1 when a parity error occurs during reception at PE=1 and is cleared when 1 is written to the CRE bit of the LIN-UART serial control register (SCR). • A reception interrupt request is outputted when this bit and the RIE bit are 1. • Data in the reception data register (RDR) is invalid when this flag is set.
bit14	ORE: Overrun error flag bit	• This bit is set to 1 when an overrun error occurs during reception and is cleared when 1 is written to the CRE bit of the LIN-UART serial control register (SCR). • A reception interrupt request is outputted when this bit and the RIE bit are 1. • Data in the reception data register (RDR) is invalid when this flag is set.
bit13	FRE: Framing error flag bit	• This bit is set to 1 when a framing error occurs during reception and is cleared when 1 is written to the CRE bit of the LIN-UART serial control register (SCR). • A reception interrupt request is outputted when this bit and the RIE bit are 1. • Data in the reception data register (RDR) is invalid when this flag is set.
bit12	RDRF: Receive data full flag bit	• This flag indicates the status of the reception data register (RDR). • This bit is set to 1 when reception data is loaded into RDR and can only be cleared to 0 when the reception data register (RDR) is read. • A reception interrupt request is outputted when this bit and the RIE bit are 1.
bit11	TDRE: Transmission data empty flag bit	• This flag indicates the status of the transmission data register (TDR). • This bit is cleared to 0 when transmission data is written to TDR and indicates that valid data exists in TDR. This bit is set to 1 when data is loaded into the transmission shift register and transmission start and indicates that no valid data exists in TDR. • A transmission interrupt request is generated if both this bit and the TIE bit are 1. • If the LBR bit in the ECCR register is set to "1" while the TDRE bit is "1", then this bit once changes to "0". After the completion of LIN synch break generator, the TDRE bit changes back to "1". Note: This bit is set to 1 (TDR empty) as its initial value.
bit10	BDS: Transfer direction selection bit	• This bit selects whether to transfer serial data from the least significant bit (LSB first, BDS=0) or the most significant bit (MSB first, BDS=1). Note: The high-order and low-order sides of serial data are interchanged with each other during reading from or writing to the serial data register. If this bit is set to another value after the data is written to the RDR register, the data becomes invalid. This bit is fixed to "0" in mode 3 (LIN).
bit9	RIE: Reception interrupt request enable bit	• This bit enables or disables the reception interrupt request output to the CPU. • If any of the RDRF, PE, ORE and FRE bits is set to "1" and this bit is "1", then a reception interrupt request is outputted.
bit8	TIE: Transmission request interrupt enable bit	• This bit enables or disables the transmission interrupt request output to the CPU. • A transmission interrupt request is outputted when this bit and the TDRE bit are 1.

20.4.4 Reception and Transmission Data Register (RDR/TDR)

Both RDR and TDR registers are located at the same address. At reading, it functions as the reception data register. At writing, it functions as the transmission data register.

■ Reception Data Register (RDR)

Figure 20.4-5 Transmission and Reception Data Registers (RDR/TDR)



R/W	Data register
Read	Read from reception data register
Write	Write to transmission data register

RDR is the data buffer register for serial data reception. The serial data signal transmitted to the SINn pin is converted in the shift register and stored in RDR register. When the data length is 7 bits, the uppermost bit (RDR: D7) contains 0. When the data is stored in this register and the reception data full flag bit (SSR: RDRF) is set to 1. If a reception interrupt request is enabled (SSR: RIE=1) at this point, a reception interrupt occurs.

Read RDR when the RDRF bit of the serial status register (SSR) is 1. The RDRF bit is cleared automatically to 0 when RDR is read. Also the reception interrupt is cleared if it is enabled and no error has occurred.

Data in RDR is invalid when a reception error occurs (SSR: PE, ORE, or FRE = 1).

■ Transmission Data Register (TDR)

TDR is the data buffer register for serial data transmission. When data to be transmitted is written to the transmission data register (TDR) in transmission enable state (SCR: TXE=1), it is transferred to the transmission shift register, then converted to serial data, and transmitted from the serial data output pin (SOTn pin). If the data length is 7 bits, the uppermost bit (TDR: D7) is invalid data.

When transmission data is written to this register, the transmission data empty flag bit (SSR: TDRE) is cleared to 0. When transfer to the transmission shift register is completed and transmission starts, the bit is set to 1. When the TDRE bit is 1, the next part of transmission data can be written. If transmission interrupt requests have been enabled, a transmission interrupt is generated. Write the next part of transmission data when a transmission interrupt is generated or the TDRE bit is 1.

Note:

TDR is a write-only register and RDR is a read-only register. These registers are located in the same address, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

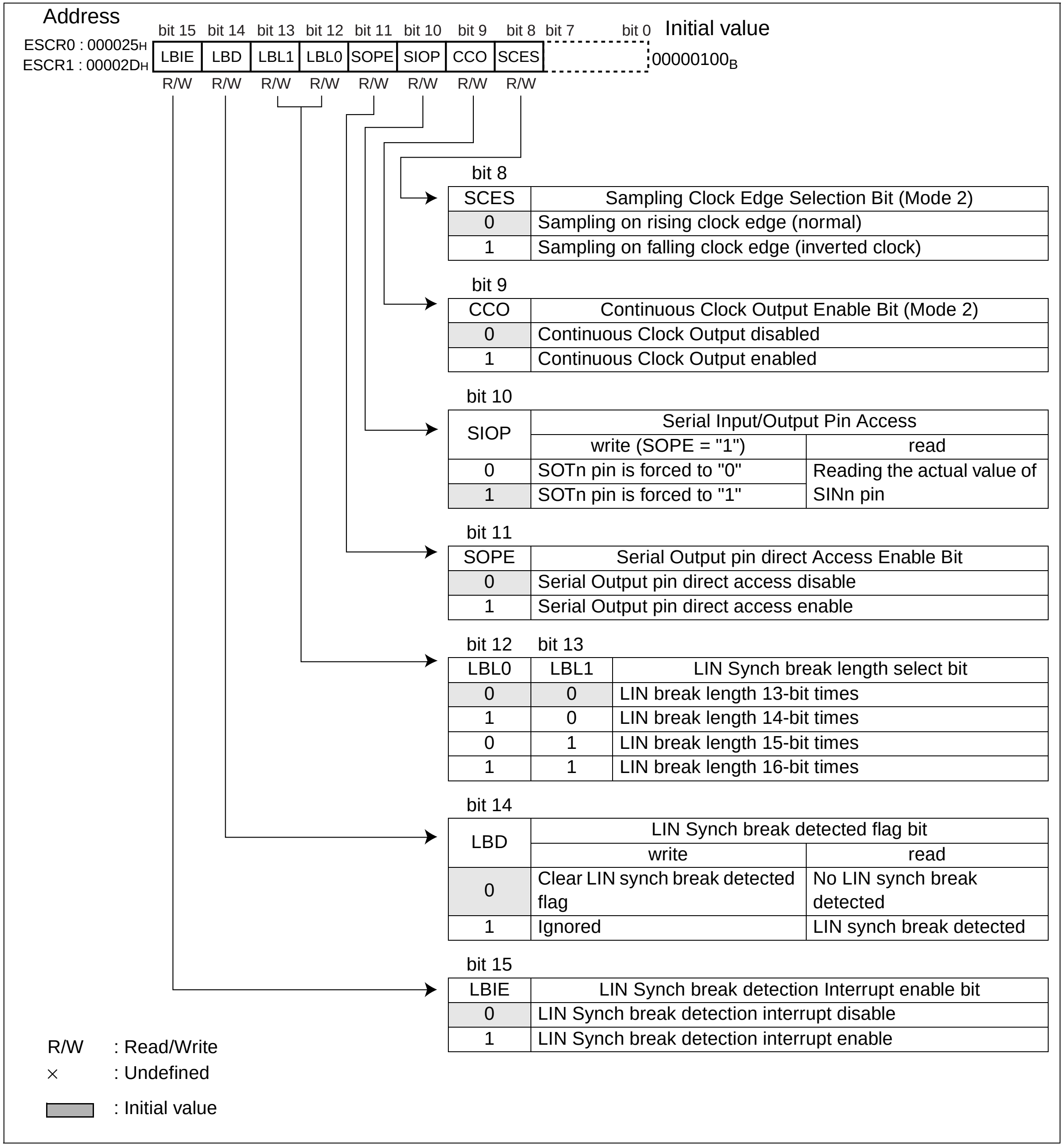
20.4.5 Extended Status/Control Register (ESCR)

This register provides several LIN functions, direct access to the SINn and SOTn pins and setting of continuous clock output and sampling clock edge in LIN-UART synchronous clock mode.

■ Extended Status/control Register (ESCR)

Figure 20.4-6 shows the Configuration of the extended status/control register (ESCR), and Table 20.4-4 shows the function of each bit.

Figure 20.4-6 Configuration of the Extended Status/control Register (ESCR)



Address	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7 … bit 0	Initial value
ESCR0 : 000025$_H$ ESCR1 : 00002D$_H$	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES		00000100$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

bit 8

SCES	Sampling Clock Edge Selection Bit (Mode 2)
0	Sampling on rising clock edge (normal)
1	Sampling on falling clock edge (inverted clock)

bit 9

CCO	Continuous Clock Output Enable Bit (Mode 2)
0	Continuous Clock Output disabled
1	Continuous Clock Output enabled

bit 10

SIOP	Serial Input/Output Pin Access	
	write (SOPE = "1")	read
0	SOTn pin is forced to "0"	Reading the actual value of SINn pin
1	SOTn pin is forced to "1"	

bit 11

SOPE	Serial Output pin direct Access Enable Bit
0	Serial Output pin direct access disable
1	Serial Output pin direct access enable

bit 12 LBL0	bit 13 LBL1	LIN Synch break length select bit
0	0	LIN break length 13-bit times
1	0	LIN break length 14-bit times
0	1	LIN break length 15-bit times
1	1	LIN break length 16-bit times

bit 14

LBD	LIN Synch break detected flag bit	
	write	read
0	Clear LIN synch break detected flag	No LIN synch break detected
1	Ignored	LIN synch break detected

bit 15

LBIE	LIN Synch break detection Interrupt enable bit
0	LIN Synch break detection interrupt disable
1	LIN Synch break detection interrupt enable

R/W	: Read/Write
×	: Undefined
▨	: Initial value

Table 20.4-4 Function in Each Bit of the Extended Status/control Register (ESCR)

NO.	Bit name	Function
bit15	LBIE: LIN synch break detection interrupt enable bit	This bit enables/disables LIN synch break detection interrupt. When the LBD bit is set to 1 and this bit is "1", a interrupt is generated. This bit is fixed to "0" in operation mode 1 and 2.
bit14	LBD: LIN synch break detected flag bit	This bit goes 1 if a LIN synch break was detected in operating mode 3. Writing a 0 to it clears this bit and the corresponding interrupt, if it is enabled. Read-modify-write instructions always return 1. Note that this dose not indicate a LIN synch break detection. Note: When LIN synch break detection is performed, disable reception (SCR: RXE=0) after enable LIN synch break detection interrupt (LBIE=1).
bit13, bit12	LBL1/0: LIN synch break length selection bits	These two bits determine how many serial bit times the LIN synch break is generated by LIN-UART. Receiving a LIN synch break is always fixed to 11 bit times.
bit11	SOPE: Serial Output pin direct access enable bit *	Setting this bit to 1 enables the direct write to the SOTn pin, if SOE = 1 (SMR). *
bit10	SIOP: Serial Input/Output Pin direct access bit *	Normal read instructions always return the actual value of the SINn pin. Writing to it sets the bit value to the SOTn pin, if SOPE = 1. During a Read-Modify-Write instruction the bit returns the SOTn value in the read cycle. *
bit9	CCO: Continuous Clock Output enable bit	This bit enables a continuous serial clock output at the SCKn pin if LIN-UART operates in master operation mode 2 (synchronous) and the SCKn pin is configured as a clock output. <Note> When CCO bit is "1", use SSM bit of ECCR as setting to "1".
bit8	SCES: Sampling clock edge selection bit	This bit inverts the serial clock signal in operation mode 2 (synchronous communication). Receiving data is sampled at the falling edge of the internal clock. If the MS bit of the ECCR register is "0" (master mode) and the SCKE bit of the SMR register is "1" (clock output enabled), the output clock signal is also inverted. During operation mode 0,1,3, please set this bit to 0.

*: Refer to the following table.

Table 20.4-5 Description of the Interaction of SOPE and SIOP

SOPE	SIOP	Writing to SIOP	Reading from SIOP
0	R/W	Has no effect on SOTn, but holds the written value	Returns current value of SINn
1	R/W	Write "0" or "1" to SOTn	Returns current value of SINn
1	RMW	Reads current value of SOTn and write it back	

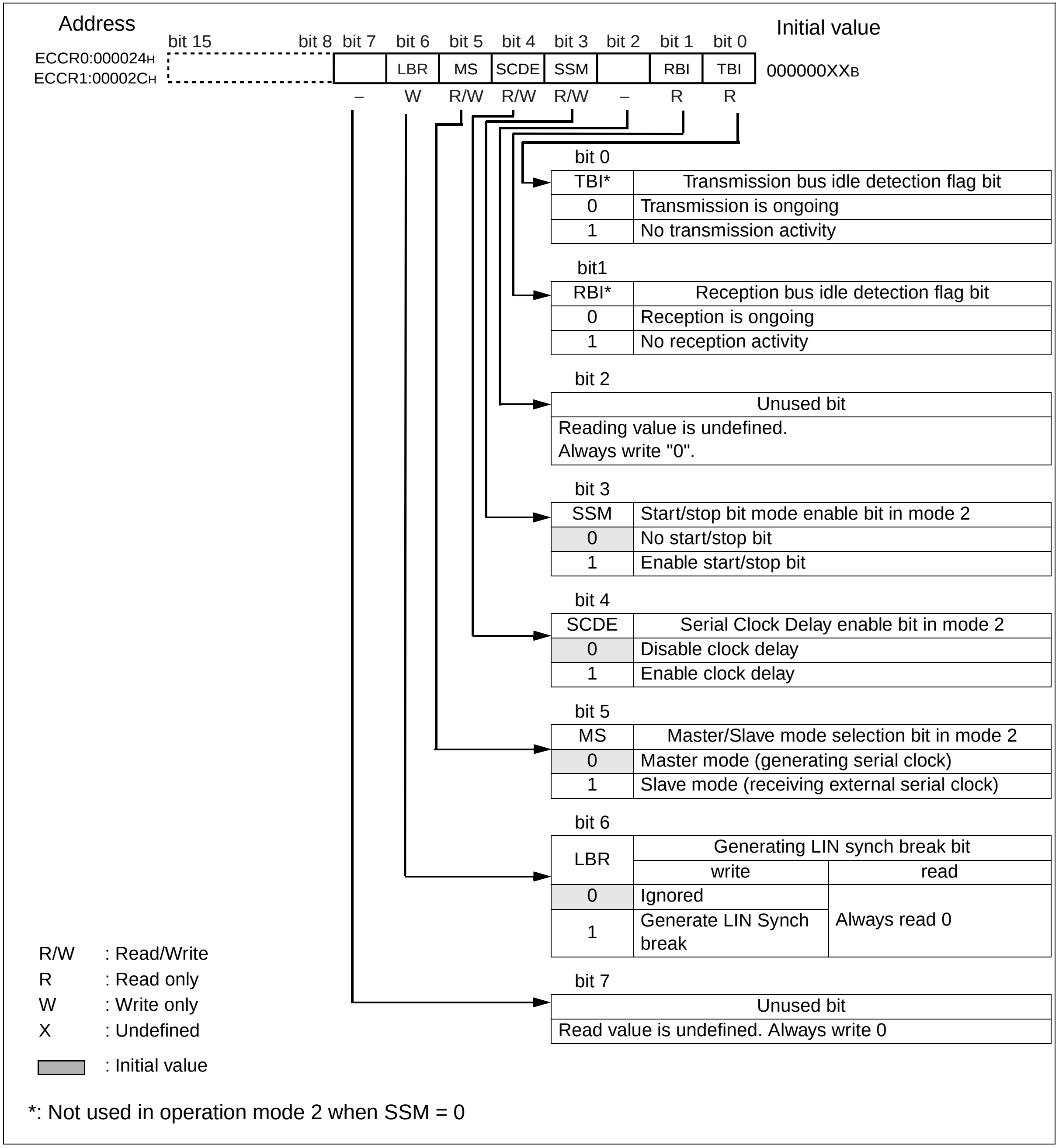
20.4.6 Extended Communication Control Register (ECCR)

The extended communication control register (ECCR) provides bus idle detection, synchronous clock settings, and the LIN synch break generation.

■ Extended Communication Control Register (ECCR)

Figure 20.4-7 shows the configuration of the extended communication control register (ECCR), and Table 20.4-6 shows the function of each bit.

Figure 20.4-7 Configuration of the Extended Communication Control Register (ECCR)



bit 0

TBI*	Transmission bus idle detection flag bit
0	Transmission is ongoing
1	No transmission activity

bit1

RBI*	Reception bus idle detection flag bit
0	Reception is ongoing
1	No reception activity

bit 2

Unused bit
Reading value is undefined. Always write "0".

bit 3

SSM	Start/stop bit mode enable bit in mode 2
0	No start/stop bit
1	Enable start/stop bit

bit 4

SCDE	Serial Clock Delay enable bit in mode 2
0	Disable clock delay
1	Enable clock delay

bit 5

MS	Master/Slave mode selection bit in mode 2
0	Master mode (generating serial clock)
1	Slave mode (receiving external serial clock)

bit 6

LBR	Generating LIN synch break bit	
	write	read
0	Ignored	Always read 0
1	Generate LIN Synch break	

bit 7

Unused bit
Read value is undefined. Always write 0

R/W : Read/Write
R : Read only
W : Write only
X : Undefined

▨ : Initial value

*: Not used in operation mode 2 when SSM = 0

Table 20.4-6 Function of Each Bit in the Extended Communication Control Register (ECCR)

NO.	Bit name	Function
bit7	Unused bit	This bit is unused bit. Reading bit is undefined. Always write "0".
bit6	LBR: Lin Synch break Generating bit	Writing a 1 to this bit generates a LIN synch break of the length selected by the LBL0/LBL1 bits of the ESCR, if operation mode 3 is selected. Setting to "0" in operation mode 0.
bit5	MS: Master/Slave mode selection bit in mode 2	This bit selects master or slave mode of LIN-UART in synchronous mode 2. If master is selected LIN-UART generates the synchronous clock by itself. If slave mode is selected, LIN-UART receives external serial clock. This bit is fixed to "0" in operation mode 0, 1 and 3. Change this bit, when the SCR: TXE bit is "0". Note: If slave mode is selected, the clock source must be external and enabled the external clock input (SMR: SCKE = 0, EXT = 1, OTO = 1).
bit4	SCDE: Serial clock delay enable bit in mode 2	If this bit is set to 1 the serial output clock is delayed as shown in Figure 20.7-5 if LIN-UART operates in master mode 2. This bit is enabled to SPI. This bit is fixed to "0" in operation mode 0, 1, and 3.
bit3	SSM: Start/Stop bit mode enable bit in mode 2	This bit adds start and stop bits to the synchronous data format in operation mode 2. It is ignored in mode 0, 1, and 3. This bit is fixed to "0" in operation mode 0, 1, and 3.
bit2	Unused bit	Unused bit. Reading value is undefined. Always write to "0".
bit1	RBI: Reception bus idle detection flag bit	This bit is "1" if there is no reception activity on the SINn pin and it is kept at "H". Do not use this bit in mode 2 when SSM=0.
bit0	TBI: Transmission bus idle detection flag bit	This bit is "1" if there is no transmission activity on the SOTn pin. Do not use this bit in mode 2 when SSM=0.

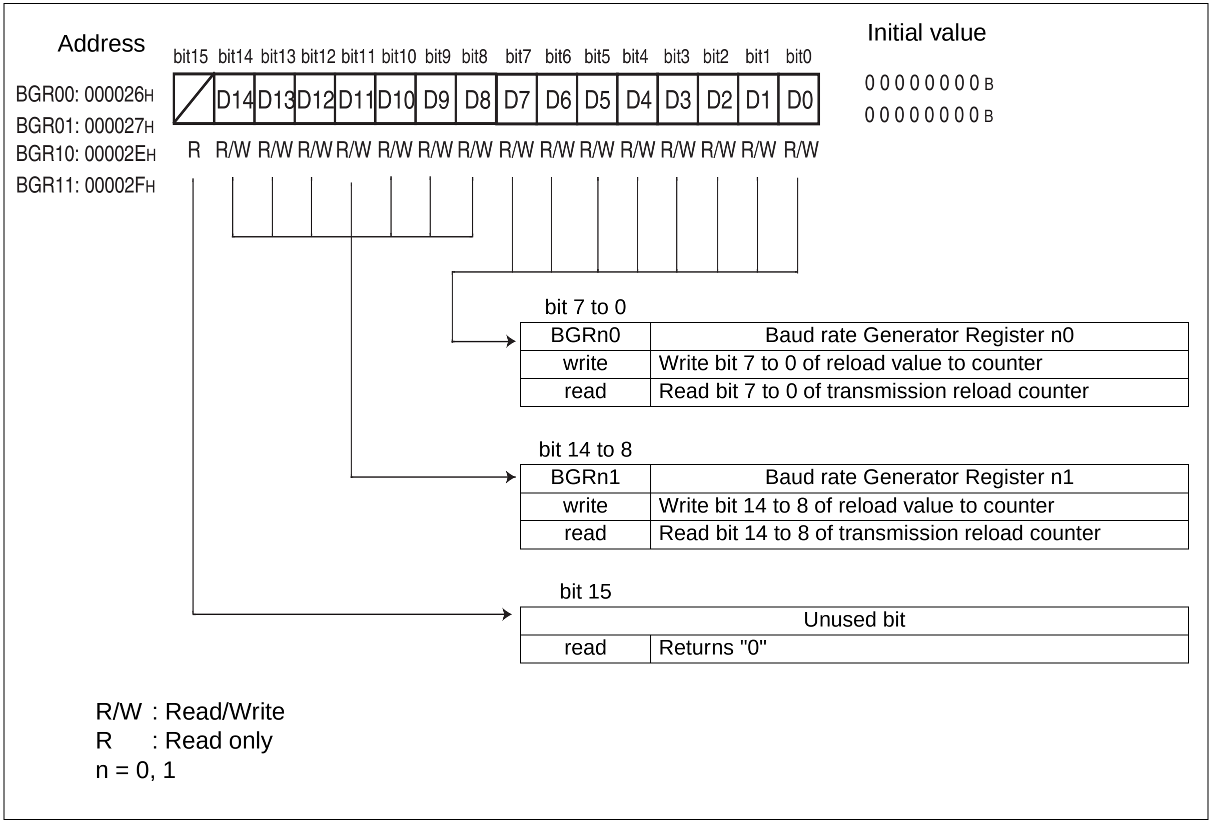
20.4.7 Baud Rate Generator Register 0 and 1 (BGR0/1)

The baud rate generator registers set the division ratio for the serial clock. Also, the actual count of the transmission reload counter can be read.

■ Baud Rate Generator Register (BGRn0/n1)

Figure 20.4-8 shows the configuration of the baud rate generator register (BGRn0/n1).

Figure 20.4-8 Configuration of Baud Rate Generator Register (BGRn0/n1)



Address

BGR00: 000026H
BGR01: 000027H
BGR10: 00002EH
BGR11: 00002FH

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value

00000000B
00000000B

bit 7 to 0

BGRn0	Baud rate Generator Register n0
write	Write bit 7 to 0 of reload value to counter
read	Read bit 7 to 0 of transmission reload counter

bit 14 to 8

BGRn1	Baud rate Generator Register n1
write	Write bit 14 to 8 of reload value to counter
read	Read bit 14 to 8 of transmission reload counter

bit 15

Unused bit	
read	Returns "0"

R/W : Read/Write
R : Read only
n = 0, 1

The baud rate generator registers determine the division ratio for the serial clock.

The BGRn1 corresponds to the upper bit and BGRn0 to lower bit, writing of the reload value to counter and reading of the transmission reload counter value are allowed. Also, byte and word access are enabled. When writing the reload value to the baud rate generator register, the reload counter starts counting.

20.5 LIN-UART Interrupts

LIN-UART uses both reception and transmission interrupts. An interrupt request can be generated for either of the following causes:

- **Receive data is set in the reception data register (RDR), or a reception error occurs.**
- **Transmission data is transferred from the transmission data register (TDR) to the transmission shift register and transmission is started.**
- **A LIN break is detected.**

The extended intelligent I/O service (EI^{2}OS) is available for these interrupts.

■ LIN-UART Interrupts

Table 20.5-1 shows the interrupt control bits and interrupt cause of the LIN-UART.

Table 20.5-1 Interrupt Control Bits and Interrupt Cause of LIN-UART

Reception/ transmission /ICU	Interrupt request flag bit	Flag register	Operation mode				Interrupt cause	Interrupt cause enable bit	How to clear the interrupt request
			0	1	2	3			
Reception	RDRF	SSR	❍	❍	❍	❍	Receive data is written to RDR.	SSR:RIE	Receive data is read.
	ORE	SSR	❍	❍	❍	❍	Overrun error		"1" is written to clear reception error flag bit (SCR: CRE).
	FRE	SSR	❍	❍	Δ	❍	Framing error		
	PE	SSR	❍	×	Δ	×	Parity error		
	LBD	ESCR	×	×	×	❍	LIN Synch break detected	ESCR:LBIE	"0" is written to ESCR: LBD.
Transmission	TDRE	SSR	❍	❍	❍	❍	TDR empty	SSR:TIE	Write data to TDR
Input Capture	ICP0/ICP1	ICS01	×	×	×	❍	1st falling edge of LIN synch field	ICS01: ICE0/ICE1	Disable ICP0/ ICP1 temporary
	ICP0/ICP1	ICS01	×	×	×	❍	5th falling edge of LIN synch field		

❍: Used bit

×: Unused bit

Δ: Only available if ECCR/SSM = 1

● Reception Interrupt

If one of the following events occurs in reception mode, the corresponding flag bit of the serial status register (SSR) is set to "1":

- Data reception is completed, i. e. the received data was transferred from the serial input shift register to the reception data register (RDR) and data can be read: RDRF
- Overrun error, i. e. RDRF = 1 and RDR was not read by the CPU and next serial data is received: ORE
- Framing error, i. e. a stop bit was expected, but a "0"-bit was received: FRE
- Parity error, i. e. a wrong parity bit was detected: PE

If at least one of these flag bits above go "1" and the reception interrupt is enabled (SSR: RIE = 1), a reception interrupt request is generated.

If the reception data register (RDR) is read, the RDRF flag is automatically cleared to "0". The error flags are cleared to "0", if a "1" is written to the clear reception error (CRE) flag bit of the serial control register (SCR).

Note:

The CRE flag is "write only" and by writing a "1" to it, it is internally held to "1" for one clock cycle.

● Transmission Interrupt

If transmission data is transferred from the transmission data register (TDR) to the transfer shift register and transfer is started, the transmission data empty flag bit (TDRE) of the serial status register (SSR) is set to "1". In this case an interrupt request is generated, if the transmission interrupt enable (TIE) bit of the SSR was set to "1" before.

Note:

The initial value of TDRE (after hardware or software reset) is "1". So an interrupt is generated immediately then, if the TIE flag is set to "1". Also note, that the only way to reset the TDRE flag is writing data to the transmission data register (TDR).

● LIN Synchronization Break Interrupt

This paragraph is only relevant, if LIN-UART operates in mode 3 as a LIN slave.

If the bus (serial input) goes "0" (dominant) for more than 11 bit times, the LIN synch break detected (LBD) flag bit of the extended status/control register (ESCR) is set to "1". Note, that in this case after 9 bit times the reception error flags are set to "1", therefore the RXE flag has to set to "0", if only a LIN synch break detect is desired.

The LIN synch break interrupt and the LBD flag are cleared after writing a "0" to the LBD flag. The LBD flag has to be performed before input capture interrupt for LIN synch field.

When LIN synch break detection is performed, it is necessary to disable the reception (SCR: RXE=0).

● LIN Synchronization Field Edge Detection Interrupts

This paragraph is only relevant, if LIN-UART operates in mode 3 as a LIN slave. After LIN synch break detection, the internal signal is set to "1" at first falling edge of the LIN synch field and to "0" after fifth falling edge. When the internal signal is set in the capture side to be inputted to capture (ICV0/1) and to be detected both edges, the interrupt occurs if the capture interrupt is enabled. The difference of the count values detected in the capture is serial clock 8 bits for master, and new baud rate can be calculated.

When the falling edge of the start bit is detected, the reload counter restarts automatically.

■ LIN-UART Interrupts and EI^{2}OS

Table 20.5-2 LIN-UART Interrupts and EI^{2}OS

Channel	Interrupt number	Interrupt control register		Vector table address			EI^{2}OS
		Register name	Address	Lower	Upper	Bank	
LIN-UART0 reception	#35(23$_H$)	ICR12	0000BC$_H$	FFFF70$_H$	FFFF71$_H$	FFFF72$_H$	*1
LIN-UART0 transmission	#36(24$_H$)	ICR12	0000BC$_H$	FFFF6C$_H$	FFFF6D$_H$	FFFF6E$_H$	*2
LIN-UART1 reception	#37(25$_H$)	ICR13	0000BD$_H$	FFFF68$_H$	FFFF69$_H$	FFFF6A$_H$	*1
LIN-UART1 transmission	#38(26$_H$)	ICR13	0000BD$_H$	FFFF64$_H$	FFFF65$_H$	FFFF66$_H$	*2

*1: Usable when ICR12 and ICR13 or interrupt causes that share an interrupt vector are not used. Provided with a function that detects a LIN-UART reception error and stops EI^{2}OS.

*2: Usable when ICR12 and ICR13 or interrupt causes that share an interrupt vector are not used.

■ LIN-UART EI^{2}OS functions

LIN-UART has a circuit for operating EI^{2}OS, which can be started up for either reception or transmission interrupts.

● For Reception

EI^{2}OS can be used if other interrupt is not enabled because the UART shares the interrupt control registers with transmission interrupt and other UART.

● For Transmission

LIN-UART shares the interrupt control registers with the LIN-UART reception interrupts and other UART. Therefore, EI^{2}OS can be started up only when no LIN-UART reception interrupts are used.

20.5.1 Reception Interrupt Generation and Flag Set Timing

The following are the reception interrupt causes: completion of reception (SSR: RDRF) and occurrence of a reception error (SSR: PE, ORE, or FRE).

■ Reception Interrupt Generation and Flag Set Timing

The received data is stored in the RDR register if the first stop bit is detected in mode 0, 1, 2 (if SSM = 1), 3, or the last data bit was read in mode 2 (if SSM = 0).

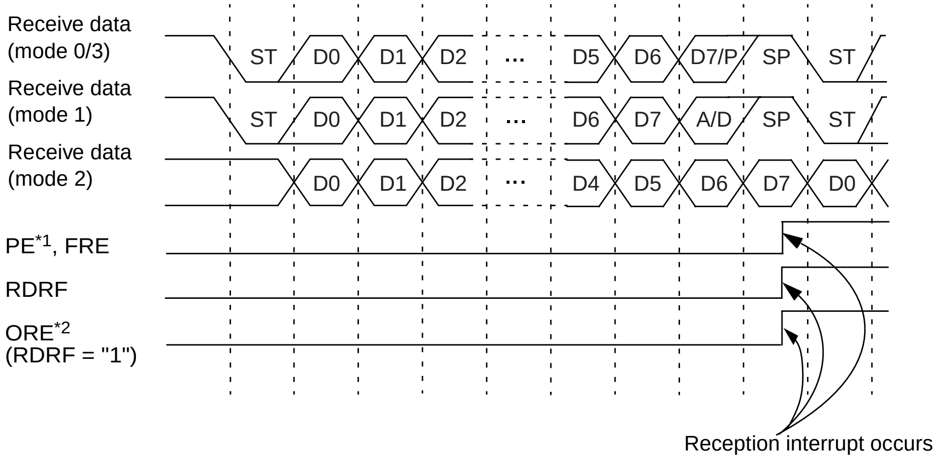
Each flag is set if the received data is completed (RDRF = 1) and the reception error (PE, ORE, FRE) of the Serial Status Register (SSR) was set to "1". In this case, if the reception interrupt is enabled (SSR: RIE=1), reception interrupt occurs.

Note:

If a reception error has occurred, the Reception Data Register (RDR) contains invalid data in each mode.

Figure 20.5-1 shows the reception operation and flag set timing.

Figure 20.5-1 Reception Operation and Flag Set Timing



*1: The PE flag will always remain "0" in mode 1 or 3.

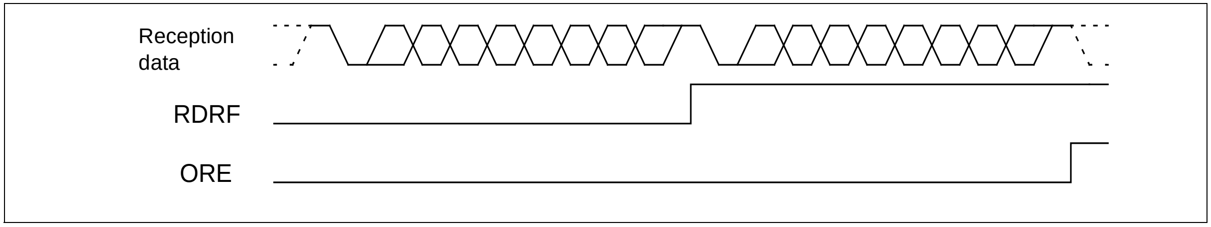
*2: ORE only occurs, if next data is transferred before the reception data is read (RDRF=1).

ST: Start Bit SP: Stop Bit A/D: Mode 1 (multiprocessor) address/data selection bit

Note:

The example in Figure 20.5-1 does not show all possible reception options for mode 0. Here it is: "7p1" and "8N1" (p = "E" [even] or "O" [odd]).

Figure 20.5-2 ORE Flag Set Timing:



20.5.2 Transmission Interrupt Generation and Flag Set Timing

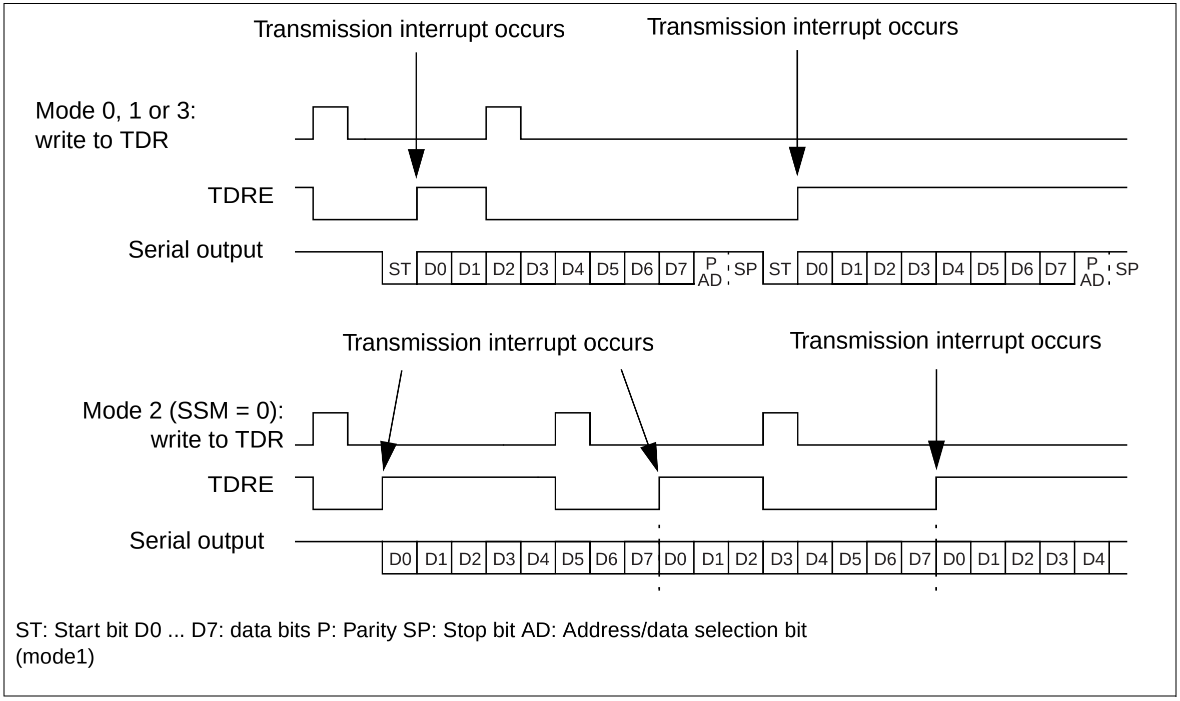
A transmission interrupt is generated when the transmission data is transferred from transmission data register (TDR) to transmission shift register and transmission is started.

■ Transmission Interrupt Generation and Flag Set Timing

When the data written to the TDR register is transferred to the transmission shift register and the transmission is started, next data to be written is enabled (SSR: TDRE=1). Then, if transmission interrupt is enabled (SSR: TIE=1), the transmission interrupt occurs. Because the TDRE bit is "read only", it only can be cleared to "0" by writing data into TDR.

The following figure demonstrates the transmission operation and flag set timing for the four modes of LIN-UART.

Figure 20.5-3 Transmission Operation and Flag Set Timing


Transmission interrupt occurs
Transmission interrupt occurs
Mode 0, 1 or 3:
write to TDR
TDRE
Serial output
ST D0 D1 D2 D3 D4 D5 D6 D7 P AD SP ST D0 D1 D2 D3 D4 D5 D6 D7 P AD SP
Transmission interrupt occurs
Transmission interrupt occurs
Mode 2 (SSM = 0):
write to TDR
TDRE
Serial output
D0 D1 D2 D3 D4 D5 D6 D7 D0 D1 D2 D3 D4 D5 D6 D7 D0 D1 D2 D3 D4
ST: Start bit D0 ... D7: data bits P: Parity SP: Stop bit AD: Address/data selection bit
(mode1)

Note:

The example in Figure 20.5-3 does not show all possible transmission options for mode 0. Here it is: "8p1" (p = "E" [even] or "O" [odd]). Parity is not provided in mode 3 or 2, if SSM = 0.

■ Transmission interrupt request generation timing

If the TDRE flag is set to "1" when a transmission interrupt is enabled (SSR: TIE=1), transmission interrupt is generated.

Note:

A transmission interrupt is generated immediately after the transmission interrupt is enabled (SSR: TIE=1) because the TDRE bit is set to 1 as its initial value. TDRE is a read-only bit that can be cleared only by writing new data to the transmission data register (TDR). Carefully specify the transmission interrupt enable timing.

20.6 LIN-UART Baud Rates

One of the following can be selected for the LIN-UART transmission/reception clock source:

- **Dedicated baud rate generator (Reload Counter)**
- **Input external clock to baud rate generator (Reload Counter)**
- **External clock (directly use SCKn pin input clock)**

■ LIN-UART Baud Rate Selection

The baud rate selection circuit is designed as shown below. One of the following three types of baud rates can be selected:

● Baud rates determined using the dedicated baud rate generator (reload counter) with internal clock

LIN-UART has two independent internal reload counters for transmission and reception serial clock. The baud rate can be selected via the 15-bit reload value determined by the Baud Rate Generator Register 0 and 1 (BGR0/1).

The reload counter divides the internal clock by set value.

It is used in asynchronous or synchronous (master) mode. Internal clock and baud rate generator clock is selected for the setting of clock source (SMR: EXT=0, OTO=0).

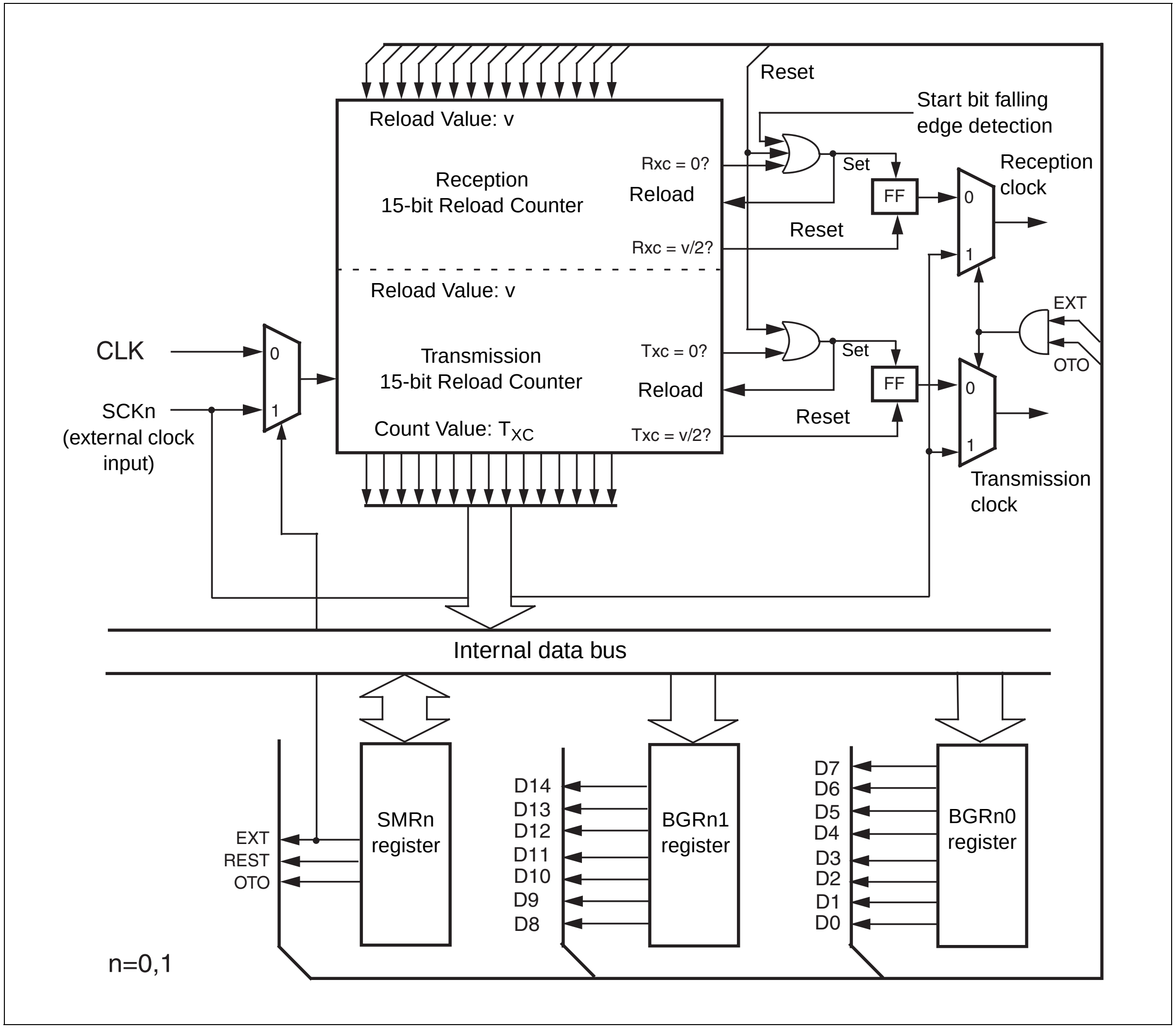
● Baud rates determined using the dedicated baud rate generator (reload counter) with external clock

An external clock source can also be connected internally to the reload counter. The baud rate can be selected via the 15-bit reload value determined by the baud rate generator register 0 and 1 (BGR 0/1). The reload counter divides the external clock by set value. It is used in asynchronous mode. External clock and baud rate generator clock is selected for the setting of the clock source (SMR: EXT=1, OTO=0). This was designed to use quartz oscillators with special frequencies and having the possibility to divide them.

● Baud rates determined using external clock (one-to-one mode)

The clock input from LIN-UART clock pulse input pins (SCKn) is used as it is (synchronous mode 2 slave operation (ECCR: MS=1)). It is used in synchronous mode (slave). External clock and direct use of external clock is selected for the setting of clock source (SMR: EXT=1, OTO=1).

Figure 20.6-1 Baud Rate Selection Circuit of LIN-UART



20.6.1 Setting the Baud Rate

This section describes how the baud rates are set and the resulting serial clock frequency is calculated.

■ Calculating the Baud Rate

The both 15-bit reload counters are programmed by the baud rate generator registers 1, 0 (BGR1/BGR0). The following calculation formula should be used to set the desired baud rate:

Reload Value:

$\mathbf{v = [\Phi / b] - 1,}$

where Φ is the machine clock, b the baud rate and [] gaussian brackets (mathematical rounding function).

● Example of calculation

If the machine clock is 16 MHz and the desired baud rate is 19200 bps baud then the reload value v is:

$\mathbf{v = [16*10^6 / 19200] - 1 = 832}$

The exact baud rate can then be recalculated: $b_{exact} = \Phi / (v + 1)$, here it is: $16*10^6 / 833 = 19207.6831$

Note:

Setting the reload value to 0 stops the reload counter. For this reason the minimum division ratio is 2. For asynchronous communication, the reload value must be greater than equal to 4 because 5 times over-sampling is performed to determine the reception value internally.

■ Suggested division ratios for different machine speeds and baud rates

The following settings are suggested for different MCU clock speeds and baud rates:

Table 20.6-1 Suggested Baud Rates and Reload Values at Different Machine Speeds.

Baud rate	8 MHz		10 MHz		16 MHz		20 MHz		24 MHz	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
4M	-	-	-	-	-	-	4	0	5	0
2M	-	-	4	0	7	0	9	0	11	0
1M	7	0	9	0	15	0	19	0	23	0
500000	15	0	19	0	31	0	39	0	47	0
460800	-	-	-	-	-	-	-	-	51	-0.16
250000	31	0	39	0	63	0	79	0	95	0
230400	-	-	-	-	-	-	-	-	103	-0.16
153600	51	-0.16	64	-0.16	103	-0.16	129	-0.16	155	-0.16
125000	63	0	79	0	127	0	159	0	191	0
115200	68	-0.64	86	0.22	138	0.08	173	0.22	207	-0.16
76800	103	-0.16	129	-0.16	207	-0.16	259	-0.16	311	-0.16
57600	138	0.08	173	0.22	277	0.08	346	-0.06	416	0.08
38400	207	-0.16	259	-0.16	416	0.08	520	0.03	624	0
28800	277	0.08	346	<0.01	554	-0.01	693	-0.06	832	-0.03
19200	416	0.08	520	0.03	832	-0.03	1041	0.03	1249	0
10417	767	<0.01	959	<0.01	1535	<0.01	1919	<0.01	2303	<0.01
9600	832	0.04	1041	0.03	1666	0.02	2083	0.03	2499	0
7200	1110	<0.01	1388	<0.01	2221	<0.01	2777	<0.01	3332	<0.01
4800	1666	0.02	2082	-0.02	3332	<0.01	4166	<0.01	4999	0
2400	3332	<0.01	4166	<0.01	6666	<0.01	8332	<0.01	9999	0
1200	6666	<0.01	8334	0.02	13332	<0.01	16666	<0.01	19999	0
600	13332	<0.01	16666	<0.01	26666	<0.01	-	-	-	-
300	26666	<0.01	-	-	-	-	-	-	-	-

Note:

Deviations are given in%.

Maximum Synchronous Baud Rate: MCU-Clock div. by 5.

■ Using external clock

If the EXT bit of the SMR is set to 1, an external clock is selected, which has to be connected to the SCKn pin. The external clock is used in the same way as the internal clock to the baud rate generator.

If One-to-one External Clock Input Mode (SMR: OTO=1) is selected the SCKn signal is directly connected to the LIN-UART serial clock inputs. This is needed for the LIN-UART synchronous mode 2 operating as slave device.

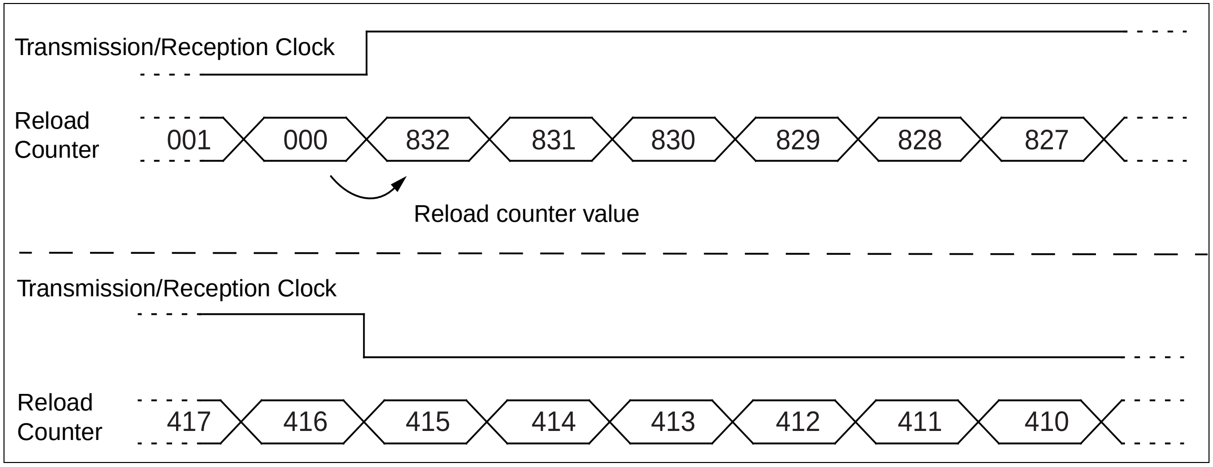
Note:

In any case the resulting clock signal is synchronized to the internal clock in the LIN-UART module. This means that indivisible clock rates will result in phase unstable signals.

■ Counting Example

Assume the reload value is 832. Figure 20.6-2 demonstrates the behavior of both Reload Counters:

Figure 20.6-2 Counting Example of the Reload Counters



Note:

The falling edge of the Serial Clock Signal always occurs after | (v + 1) / 2 |.

20.6.2 Restarting the Reload Counter

The reload counter is a 15-bit reload counter that functions as dedicated baud rate generator. The transmission/reception clock is generated by the external or internal clock. Also, the count value of the transmission reload counter can be read by the baud rate generator register (BGR1, BGR0).

■ Function of Reload Counter

The reload counter has the transmission and reception reload counters and functions as dedicated baud rate generator. It consists of a 15-bit register for the reload value and generates the transmission/reception clocks by the external or internal clock. Also, the count value of the transmission reload counter can be read by the baud rate generator register (BGR1, BGR0)

● Count start

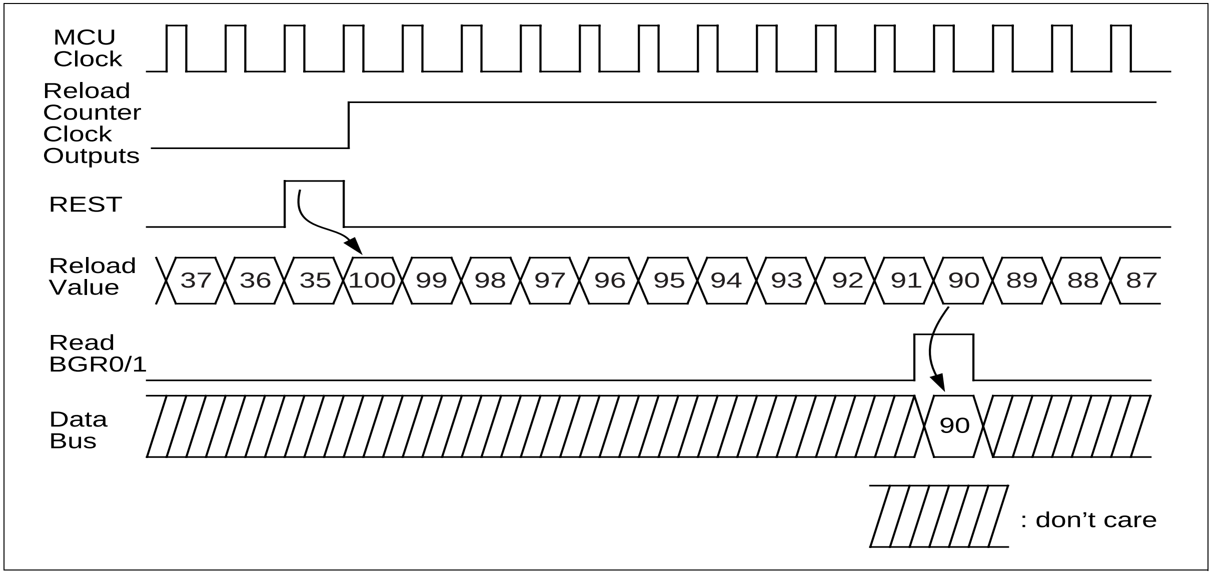
When the reload value is written to the baud rate generator register (BGR1, BGR0), the reload counter starts counting.

● Restart

If the REST bit of the Serial Mode Register (SMR) is set to "1", both Reload Counters are restarted at the next clock cycle. This feature is intended to use the Transmission Reload Counter as a simple timer.

The following figure illustrates a possible usage of this feature (assume that the reload value is 100.)

Figure 20.6-3 Reload Counter Restart Example



In this example the number of MCU clock cycles (cyc) after REST is then:

cyc = v - c + 1 = 100 - 90 + 1 = 11,

where v is the reload value and c is the read counter value.

Note:

If LIN-UART is reset by setting SMR:UPCL to "1", the Reload Counters will restart too.

- Automatic restart (reception reload counter only)
  In asynchronous LIN-UART mode, if a falling edge of a start bit is detected, the Reception Reload Counter is restarted. This is intended to synchronize the serial shift register to the incoming serial data stream.

● Clearing reload counters

The reload value of the baud rate generator register (BGR1, BGR0) and the reload counters are cleared to "00" by the MCU global reset and the counters stops. The reload counters are cleared to "00_H" by writing "1" to the UPCL bit in the SMR register. However, the value stored in the reload register is kept unchanged and the counters restart from reload value immediately. Writing "1" to the REST bit does not clear the counters and they restart from reload value immediately.

20.7 Operation of LIN-UART

LIN-UART operates in operation mode 0 for normal bidirectional serial communication, in mode 2 and 3 in bidirectional communication as master or slave, and in mode 1 as master or slave in multiprocessor communication.

■ Operation of LIN-UART

● Operation modes

There are four LIN-UART operation modes: modes 0 to 3. As listed in Table 20.7-1 , an operation mode can be selected according to the communication method.

Table 20.7-1 Operation Mode of LIN-UART

Operation mode		Data length		Synchronization of mode	Length of stop bit	Data bit format
		Parity disabled	Parity enabled			
0	Normal mode	7 or 8 bits		Asynchronous	1 or 2 bits	LSB first MSB first
1	Multiprocessor mode	7 or 8 bits + 1*	-	Asynchronous		
2	Normal mode	8		Synchronous	None, 1 or 2 bits	
3	LIN mode	8	-	Asynchronous	1 bit	LSB first

-: Setting disabled
*: "+1" means the indicator bit of the address/data selection used for controlling communication in the multiprocessor mode.

Note:

Mode 1 operation is supported both for master or slave operation of LIN-UART in a master-slave connection system. In Mode 3, the LIN-UART function is locked to 8N1-Format, LSB first.

If the mode is changed, LIN-UART cuts off all possible transmission or reception and awaits then new action.

■ Inter-CPU Connection Method

External Clock One-to-one connection (normal mode) and master-slave connection (multiprocessor mode) can be selected. For either connection method, the data length, whether to enable parity, and the synchronization method must be common to all CPUs. Select an operation mode as follows:

- In the one-to-one connection method, operation mode 0 or 2 must be used in the two CPUs. Select operation mode 0 for asynchronous transfer mode and operation mode 2 for synchronous transfer mode. Note, that one CPU has to set to the master and the other to the slave in synchronous mode 2.
- Select operation mode 1 for the master-slave connection method and use it either for the master or slave system.

■ Synchronization Methods

In asynchronous operation, LIN-UART reception clock is automatically synchronized to the falling edge of a received start bit.

In synchronous mode, the synchronization is performed either by the clock signal of the master device or by LIN-UART itself if operating as master.

■ Signal Mode

LIN-UART can treat data only in non-return to zero (NRZ) format.

■ Operation Enable Bit

LIN-UART controls both transmission and reception using the operation enable bit for transmission (SCR: TXE) and reception (SCR: RXE).

- If reception operation is disabled during reception (data is input to the reception shift register), finish frame reception and read the received data of the reception data register (RDR). Then stop the reception operation.
- If the transmission operation is disabled during transmission (data is output from the transmission shift register), wait until there is no data in the transmission data register (TDR) before stopping the transmission operation.

20.7.1 Operation in Asynchronous Mode (Op. Modes 0 and 1)

When LIN-UART is used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode), the asynchronous transfer mode is selected.

■ Operation in Asynchronous Mode

- Transfer data format

Generally each data transfer in the asynchronous mode operation begins with the start bit (low-level) and ends with at least one stop bit (high-level). The direction of the bit stream (LSB first or MSB first) is determined by the BDS bit of the Serial Status Register (SSR). The parity bit (if enabled) is always placed between the last data bit and the (first) stop bit.

In operation mode 0 the length of the data frame can be 7 or 8 bits, with or without parity, and 1 or 2 stop bits.

In operation mode 1 the length of the data frame can be 7 or 8 bits with a following address-/data-selection bit instead of a parity bit. 1 or 2 stop bits can be selected.

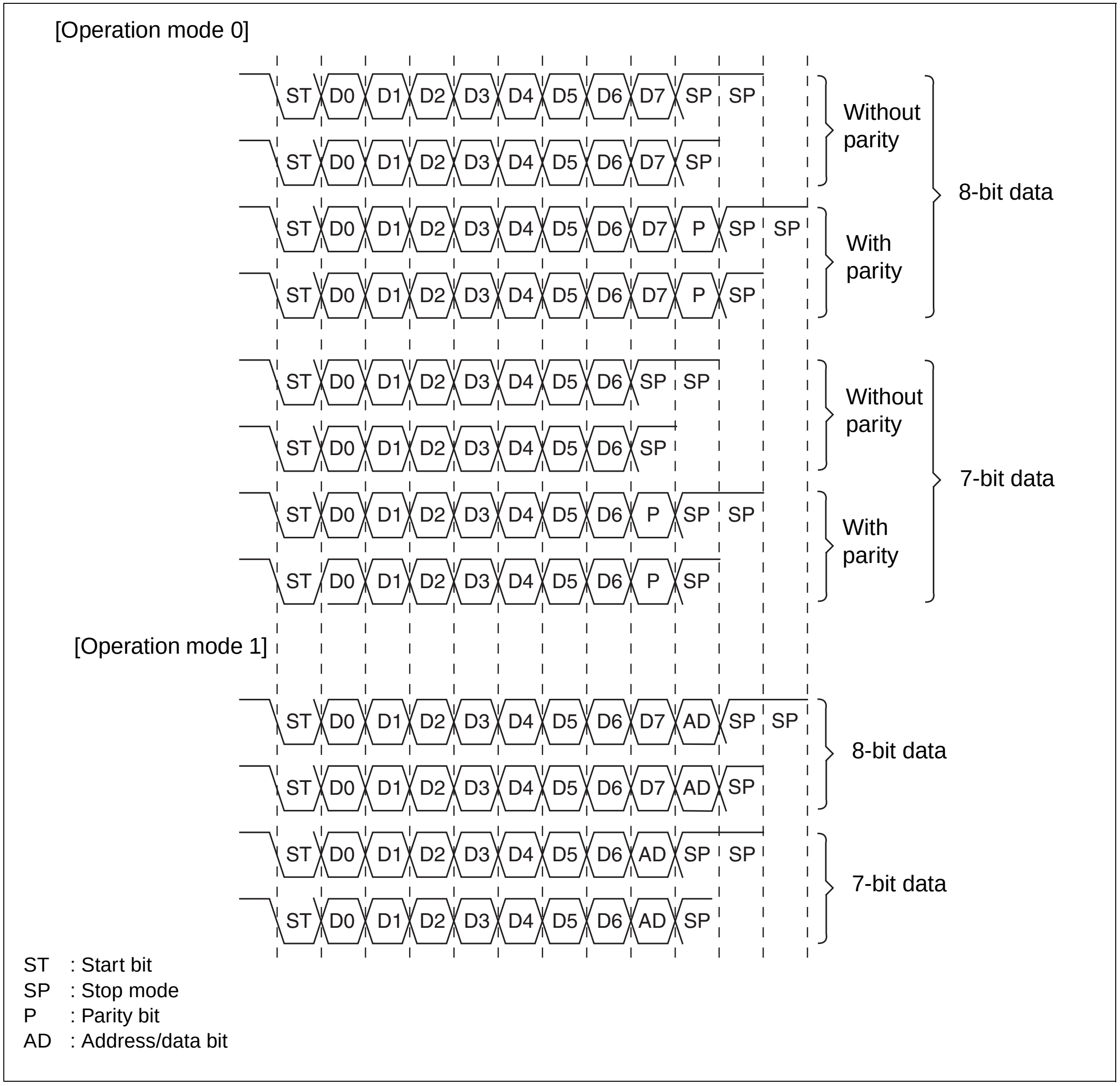
The calculation formula for the bit length of a transfer frame is:

Length = 1 + d + p + s

(d = number of data bits [7 or 8], p = parity [0 or 1], s = number of stop bits [1 or 2]

Figure 20.7-1 shows the data format in the asynchronous mode.

Figure 20.7-1 Transfer Data Format (operation Modes 0 and 1)


[Operation mode 0]
ST D0 D1 D2 D3 D4 D5 D6 D7 SP SP
ST D0 D1 D2 D3 D4 D5 D6 D7 SP
Without parity
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP SP
ST D0 D1 D2 D3 D4 D5 D6 D7 P SP
With parity
8-bit data
ST D0 D1 D2 D3 D4 D5 D6 SP SP
ST D0 D1 D2 D3 D4 D5 D6 SP
Without parity
ST D0 D1 D2 D3 D4 D5 D6 P SP SP
ST D0 D1 D2 D3 D4 D5 D6 P SP
With parity
7-bit data
[Operation mode 1]
ST D0 D1 D2 D3 D4 D5 D6 D7 AD SP SP
ST D0 D1 D2 D3 D4 D5 D6 D7 AD SP
8-bit data
ST D0 D1 D2 D3 D4 D5 D6 AD SP SP
ST D0 D1 D2 D3 D4 D5 D6 AD SP
7-bit data
ST : Start bit
SP : Stop mode
P : Parity bit
AD : Address/data bit

Note:

If BDS bit of the Serial Status Register (SSR) is set to "1" (MSB first), the bit stream processes as: D7, D6, ..., D1, D0, (P).

● Transmission operation

If the Transmission Data Register Empty (TDRE) flag bit of the Serial Status Register (SSR) is "1", transmission data is allowed to be written to the Transmission Data Register (TDR). When data is written, the TDRE flag goes "0". If the transmission operation is enabled by the TXE-Bit ("1") of the Serial Control Register (SCR), the data is written next to the transmission shift register and the transmission starts at the next clock cycle of the serial clock, beginning with the start bit.

If transmission interrupt is enabled (TIE = 1), the interrupt is generated by the TDRE flag. Note, that the initial value of the TDRE flag is "1", so that in this case if TIE is set to "1" an interrupt will occur immediately.

When the data length is set to 7 bits (CL=0), the unused bit of the TDR is always the MSB, independently from the transfer direction selection in the BDS bit (LSB first or MSB first).

Note:

As the initial value of transmission data empty flag bit (SSR: TDRE) is "1" if the transmission interrupt is enabled (SSR: TIE=1), the interrupt occurs immediately.

● Reception operation

Reception operation is performed when it is enabled by the Reception Enable (RXE) flag bit of the SCR. If a start bit is detected, a data frame is received according to the data format specified by the SCR. In case of errors, the corresponding error flags are set (SSR: PE, ORE, FRE). After the reception of the data frame, the data is transferred from the reception shift register to the Reception Data Register (RDR) and the Receive Data Register Full (RDRF) flag bit of the SSR is set to "1". In this case, if the reception interrupt request is enabled (SSR: RIE=1), the reception interrupt request is occurred. When reading data after reception of one frame data, check the error flag state and read reception data from the RDR register if the reception is performed normally. If the reception error occurs, perform error processing. The data then has to be read by the CPU. By doing so, the RDRF flag of SSR is cleared to "0".

When the data length is set to 7 bits (CL=0), the unused bit of the TDR is always the MSB, independently from the bit transfer direction selection in the BDS bit (LSB first or MSB first).

Note:

Only when the RDRF flag bit of SSR is set to "1" and no errors have occurred (SSR: PE, ORE, FRE=0) the Reception Data Register (RDR) contains valid data.

● Used clock

Use the internal clock or external clock. Select the baud rate generator (SMR: EXT = 0 or 1, OTO = 0) for desired baud rate.

● Stop bit

1- or 2-stop bit can be selected at the transmission. When 2-stop bit is selected, both stop bits is detected at the reception. When first stop bit is detected, the RDRF bit of SSR is "1". Then, when the start bit is not detected, the RBI bit of ECCR is set to "1", indicating no reception operation.

● Error detection

In mode 0, the parity, overrun, and framing errors can be detected.

In mode 1, the overrun and framing errors can be detected, and the parity error cannot be detected.

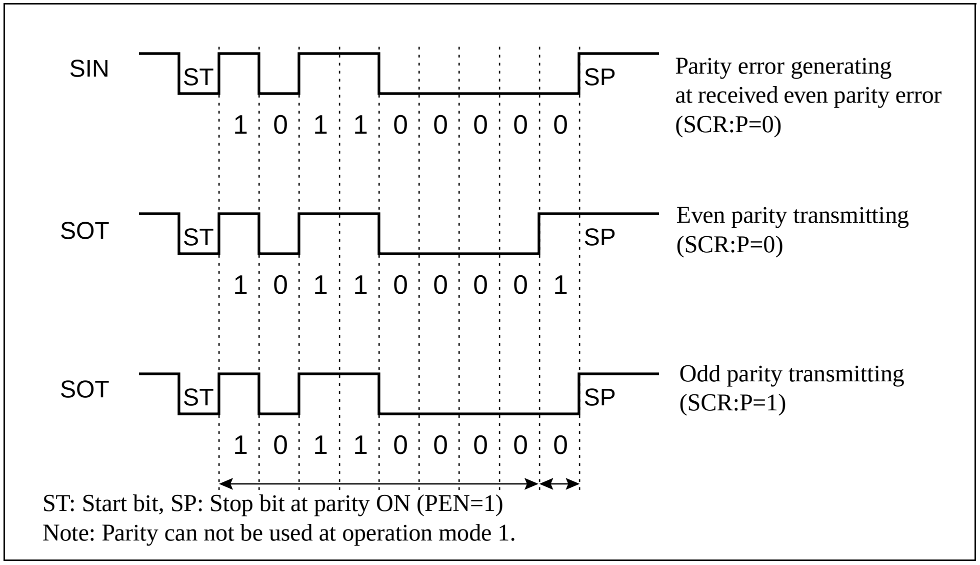
● Parity

Parity can set to add (transmission) or detect (reception) the parity bit.

The parity enable bit (SCR: PEN) is used to specify whether there is parity or not, and parity selection bit (SCR: P) is selected the even/odd parity.

In operation mode 1, the parity cannot be used.

Figure 20.7-2 Transmission Data when Parity Enabled



ST: Start bit, SP: Stop bit at parity ON (PEN=1)
Note: Parity can not be used at operation mode 1.

● Data signal type

The data signal type is NRZ data format.

● Data transition method

The data bit transfer method can be selected by LSB or MSB first.

20.7.2 Operation in Synchronous Mode (Operation Mode 2)

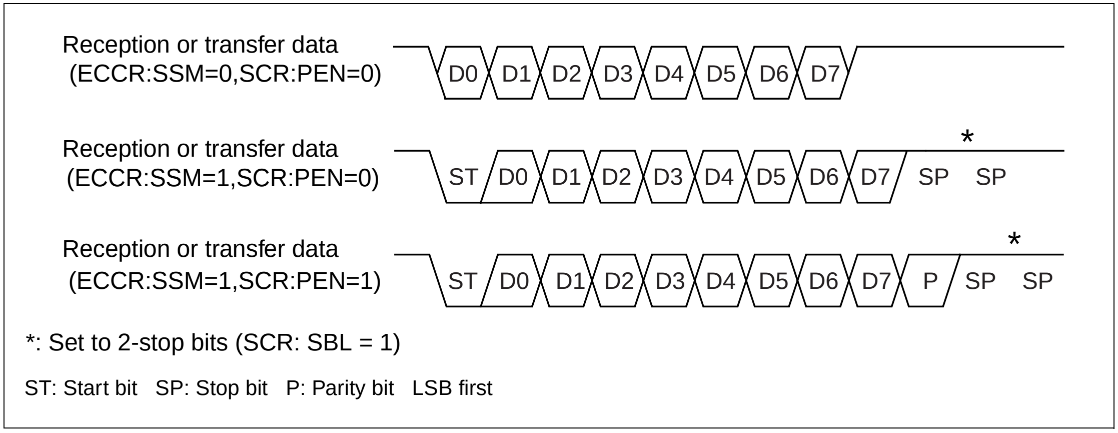
The clock synchronous transfer method is used for LIN-UART operation mode 2 (normal mode).

■ Operation in Synchronous Mode (Operation mode 2)

● Transfer data format

In the synchronous mode, 8-bit data is transferred without start or stop bits if the SSM bit of the Extended Communication Control Register (ECCR) is 0. Also, when the start/stop bit is provided (ECCR: SSM = 1), presence or absence of the parity bit can be selected (SCR: PEN). The figure below illustrates the data format during a transmission in the synchronous operation mode.

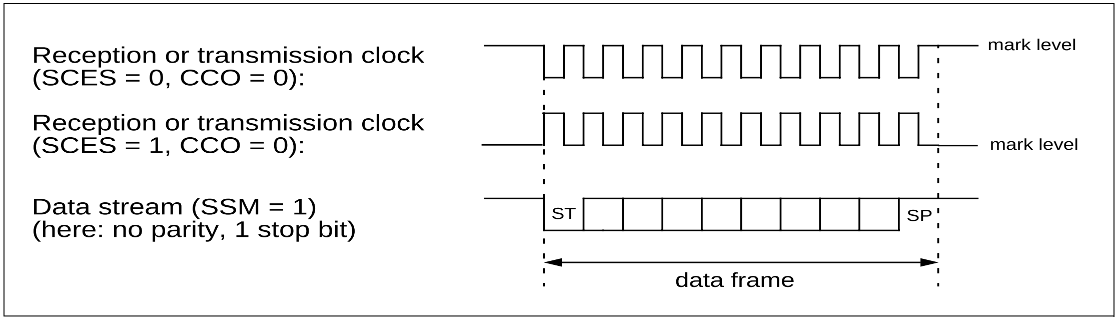
Figure 20.7-3 Transfer Data Format (operation mode 2)


Reception or transfer data
(ECCR:SSM=0,SCR:PEN=0)
D0
D1
D2
D3
D4
D5
D6
D7
Reception or transfer data
(ECCR:SSM=1,SCR:PEN=0)
ST
D0
D1
D2
D3
D4
D5
D6
D7
SP
SP
*
Reception or transfer data
(ECCR:SSM=1,SCR:PEN=1)
ST
D0
D1
D2
D3
D4
D5
D6
D7
P
SP
SP
*
*: Set to 2-stop bits (SCR: SBL = 1)
ST: Start bit SP: Stop bit P: Parity bit LSB first

● Clock inversion function

If the SCES bit of the Extended Status/Control Register (ESCR) is set to "1", the serial clock is inverted. Therefore, in slave mode LIN-UART samples the data bits at the falling edge of the received serial clock. Note, that in master mode if SCES is set to "1", the clock signal's mark level is "0".

Figure 20.7-4 Transfer Data Format with Clock Inversion



● Start/stop bits

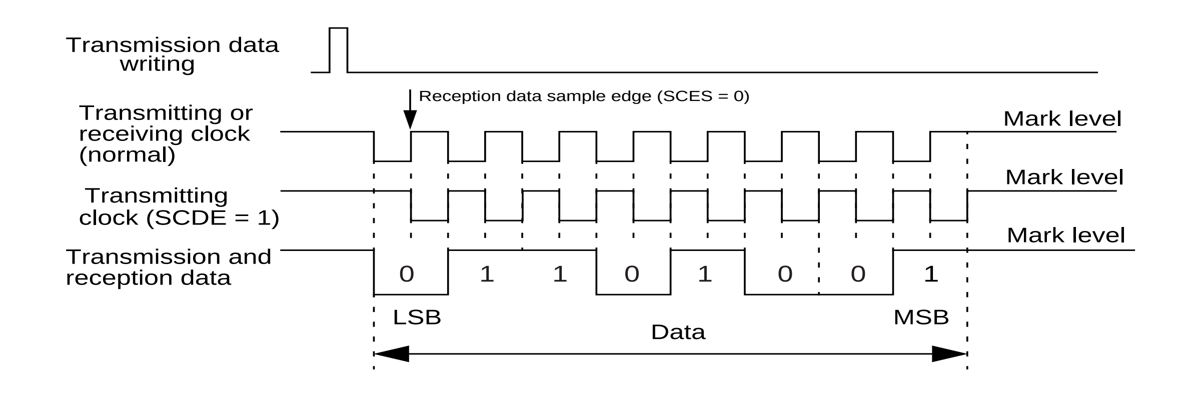
If the SSM bit of the Extended Communication Control Register (ECCR) is set to "1", the data format gets additional start and stop bits like in asynchronous mode.

- Clock supply:

In operation mode 2, the number of clock cycles for the clock signal must be the same as the number of bits for the transmission and reception. If the start/stop bits are enabled, it must be matched the additional start/ stop bits. If the MS bit of the ECCR register is "0" (master mode) and the SCKE bit of the SMR register is "1" (serial clock output enabled), the consistent clock cycles are generated automatically. If the MS bit of the ECCR register is "1" (slave mode), or if the SCKE bit of SMR is 0 (serial clock output disabled), the clock for each bit of transfer data must be supplied from outside. While there is no communication, the clock signal must be kept at "H" as the mark level

If the SCDE bit of the ECCR register is "1", the clock output signal is delayed by the half of the serial clock cycle as shown in Figure 20.7-5 . The operation is prepared for communication devices which use the rising or falling edge of the serial clock signal for the data sampling.

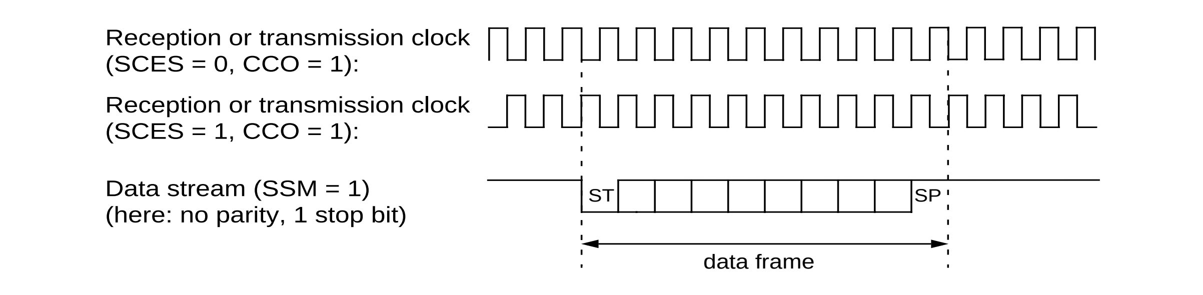
Figure 20.7-5 Delayed Transmitting Clock Signal(SCDE=1)


Transmission data writing
Reception data sample edge (SCES = 0)
Transmitting or receiving clock (normal)
Mark level
Transmitting clock (SCDE = 1)
Mark level
Transmission and reception data
Mark level
0
1
1
0
1
0
0
1
LSB
MSB
Data

If the SCES bit of the ESCR register is "1", the serial clock signal is inverted. Receiving data is sampled at the falling edge of the serial clock.

In this case, the serial data must be valid value at the falling edge of the clock. If the CCO bit of ESCR register is "1" (master mode), the serial clock output of the SCKn pin is supplied continuously. In this mode, be sure to add the start/stop bits (SSM = 1) to identify the start and end of data frame. Figure 20.7-6 shows the operation of this function.

Figure 20.7-6 Continuous Clock Supply in Mode 2


Reception or transmission clock (SCES = 0, CCO = 1):
Reception or transmission clock (SCES = 1, CCO = 1):
Data stream (SSM = 1) (here: no parity, 1 stop bit)
ST
SP
data frame

- Error detection:

If no start/stop bits are selected (ECCR: SSM = 0) only overrun errors are detected.

● Communication:

For initialization of the synchronous mode, following settings have to be done:

Baud rate generator registers (BGR0/BGR1):

Set the desired reload value for the dedicated baud rate reload counter.

Serial mode register (SMR):

MD1, MD0: "10_B" (Mode 2)

SCKE: "1" for the dedicated baud rate reload counter
"0" for external clock input

SOE: "1" for transmission and reception
"0" for reception only

Serial control register (SCR):

RXE, TXE: set one or both of these bits to "1"

AD: no address/data selection - don’t care

CL: automatically fixed to 8-bit data - don’t care

CRE: "1" to clear receive error flags. Suspend transmission and reception.

-- when SSM=0:

PEN, P, SBL: don’t care

-- when SSM=1:

PEN: "1" if parity bit is added/detected, "0" if not

P: "0" for even parity, "1" odd parity

SBL: "1" for 2 stop bits, "0" for 1 stop bit.

Serial status register (SSR):

BDS: "0" for LSB first, "1" for MSB first

RIE: "1" if reception interrupts are used; "0" reception interrupts are disabled.

TIE: "1" if transmission interrupts are used; "0" transmission interrupts are disabled.

Extended communication control register (ECCR):

SSM: "0" if no start/stop bits are desired (normal); "1" for adding start/stop bits (exteneded function)

MS: "0" for master mode (LIN-UART generates the serial clock); "1" for slave mode (LIN-UART receives serial clock from the master device)

Note:

To start the communication, write data to TDR register. When receiving the data, disable the serial output (SMR: SOE = "0") and write the dummy data to TDR. By enabling the continuous clock and start/stop bits, in the bi-directional communication the same as the asynchronous mode is allowed.

20.7.3 Operation with LIN Function (Operation Mode 3)

LIN-UART can be used either as LIN-Master or LIN-Slave. For this LIN function a special mode is provided. Setting the LIN-UART to mode 3 configures the data format to 8N1-LSB-first format.

■ Operation in Asynchronous LIN Mode (Operation mode 3)

● LIN-UART as LIN master

In LIN master mode, the master determines the baud rate of the whole bus, therefore slaves devices have to synchronize to the master. Therefore, the desired baud rate remains fixed in master operation after initialization.

Writing a "1" into the LBR bit of the Extended Communication Control Register (ECCR) generates a 13 - 16 bit time low-level on the SOTn pin, which is the LIN synchronization break and the start of a LIN message. Thereby the TDRE flag of the Serial Status Register (SSR) goes "0" and is reset to "1" after the break, and generates a transmission interrupt for the CPU (if TIE of SSR is "1").

The length of the LIN break to be sent can be determined by the LBL1/0 bits of the ESCR as follows:

Table 20.7-2 LIN Break Length

LBL0	LBL1	Break length
0	0	13 bits
1	0	14 bits
0	1	15 bits
1	1	16 bits

The Synch Field is sent as byte data of 0x55 after the LIN break. To prevent a transmission interrupt, the 0x55 can be written to the TDR just after writing the "1" to the LBR bit, although the TDRE flag is "0".

● LIN-UART as LIN slave

In LIN slave mode, LIN-UART has to synchronize to the master’s baud rate. If Reception is disabled (RXE = 0) but LIN break interrupt is enabled (LBIE = 1) LIN-UART will generate a reception interrupt, if a synchronization break from the LIN master is detected, and indicates it with the LBD flag of the ESCR to "1". Writing "0" to this bit clears the reception interrupt request. For the calculation of the baud rate, the UART0 operation is explained as an example. When UART0 detects first falling edge of synch field, set the internal signal inputted to the input capture (ICU0) to "H" and start ICU0. This internal signal is set to "L" at fifth falling edge, ICU0 must be set to the LIN mode (ICE01). Also, the ICUO interrupt must be set to enable and to detect both edges (ICS01).

The time when the ICU0 input signal is "1" is the value in which eight baud rates are multiplied. Therefore, baud rate setting value is summarized as follows:

without free-run timer overflow : BGR value = (b-a)/8 -1

with free-run timer overflow : BGR value = (Max + b-a)/8-1

where Max is the free-run timer maximum value at the overflow occurs.

where a is the value of the ICU data register after the first interrupt

where b is the value of the ICU data register after the second interrupt

Note:

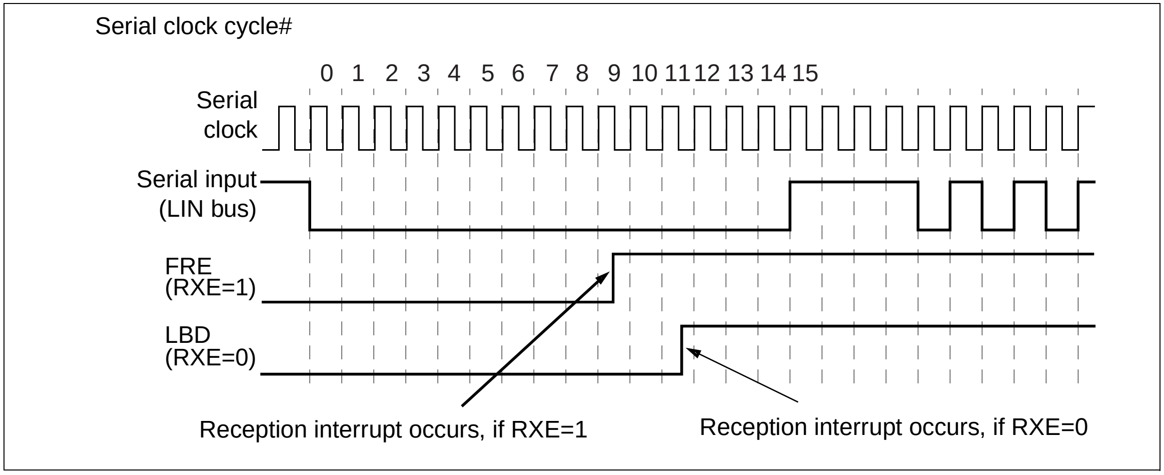
As shown in the LIN slave mode, when the BGR value newly calculated by synch field generates ±15% or more baud rate error, do not set the baud rate.

For the correspondence between other LIN-UARTs and ICUs, see "13.5 Explanation of Operation of 16-bit Free-run Timer" and "13.6 Explanation of Operation of Input Capture".

● LIN Synch Break Detection Interrupt and Flags

If a LIN Synch break is detected in the slave mode, the LIN Break Detected (LBD) Flag of the ESCR is set to "1". This causes an interrupt, if the LIN Break Interrupt Enable (LBIE) bit is set to 1.

Figure 20.7-7 LIN Synch Break Detection and Flag Set Timing.



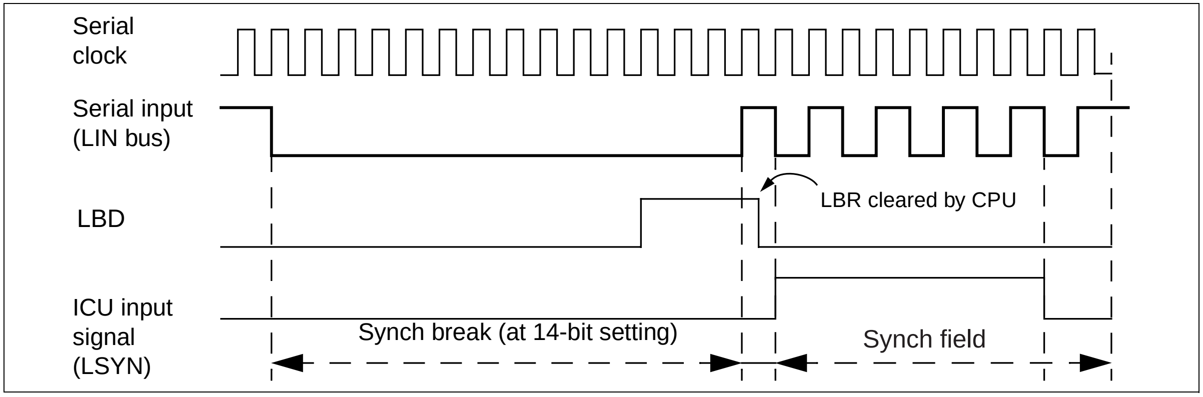
The figure above demonstrates the LIN synch break detection and flag set timing.

Note, that if reception is enabled (RXE = 1) and reception interrupt is enabled (RIE = 1) the Reception Data Framing Error (FRE) flag bit of the SSR will cause a reception interrupt 2 bit times ("8N1") earlier than the LIN break interrupt, so it is recommended to turn off RXE, if a LIN break is expected.

LIN synch break detection is only supported in operation mode 3.

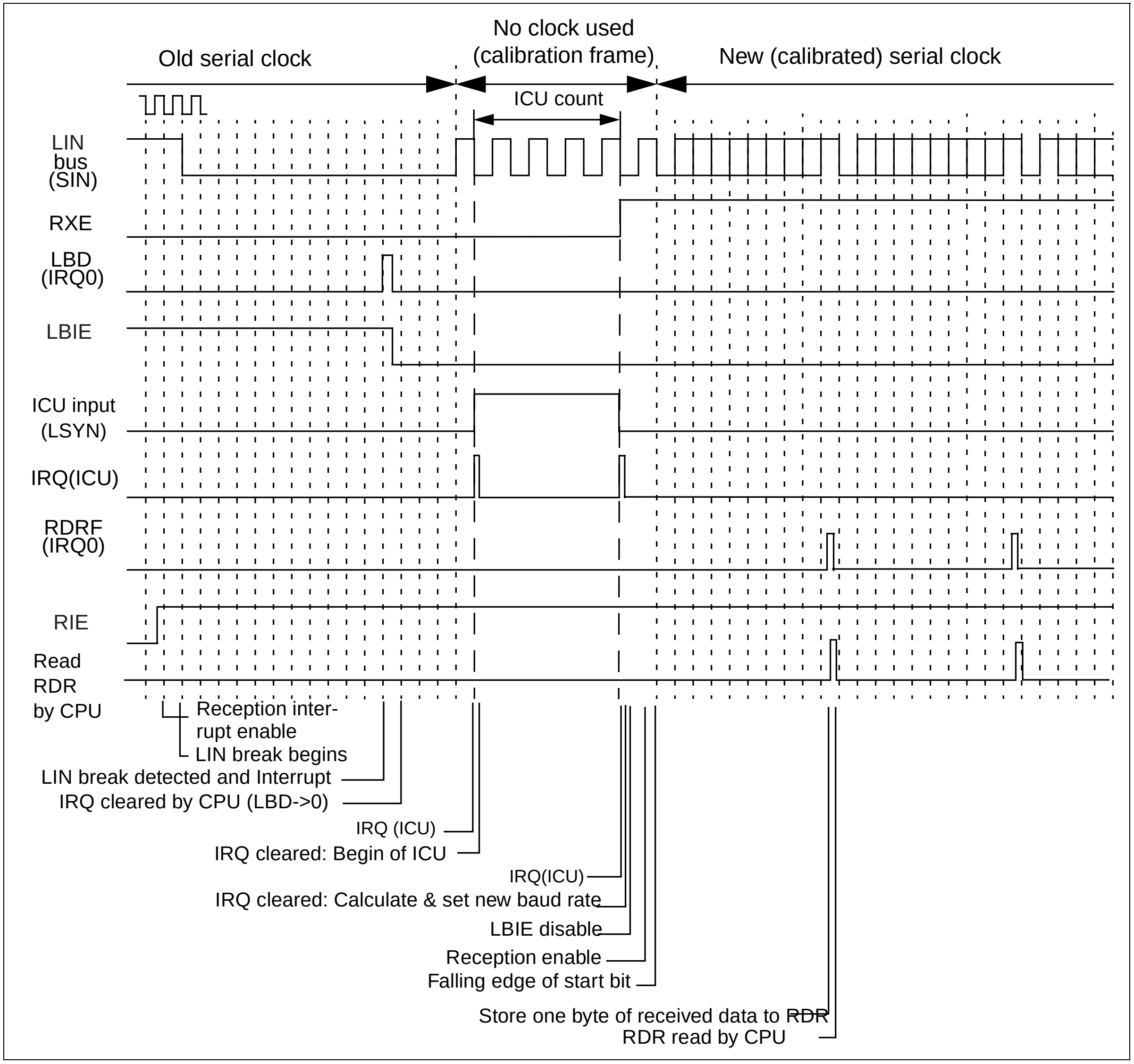
Figure 20.7-8 shows a typical start of a LIN message and the behavior of the LIN-UART.

Figure 20.7-8 LIN-UART Behavior as Slave in LIN Mode



● LIN bus timing

Figure 20.7-9 LIN Bus Timing and LIN-UART Signals


Old serial clock
No clock used (calibration frame)
New (calibrated) serial clock
ICU count
LIN bus (SIN)
RXE
LBD (IRQ0)
LBIE
ICU input (LSYN)
IRQ(ICU)
RDRF (IRQ0)
RIE
Read RDR by CPU
Reception inter-rupt enable
LIN break begins
LIN break detected and Interrupt
IRQ cleared by CPU (LBD->0)
IRQ (ICU)
IRQ cleared: Begin of ICU
IRQ(ICU)
IRQ cleared: Calculate & set new baud rate
LBIE disable
Reception enable
Falling edge of start bit
Store one byte of received data to RDR
RDR read by CPU

20.7.4 Direct Access to Serial Pins

LIN-UART allows the user to directly access to the transmission pin (SOTn) or the reception pin (SINn).

■ LIN-UART Direct Pin Access

The LIN-UART provides the ability for the software to access directly to serial input or output pin. The software can always monitor the incoming serial input pin (SINn) by reading the SIOP bit of the ESCR. If direct write to the serial output pin (SOTn) is allowed (ESCR: SOPE = 1) when the serial output is enabled (SMR: SOE = 1) after "0" or "1" is written to the SIOP bit of the ESCR register, the SOTn value can be set arbitrarily.

In LIN mode, this function can be used for reading back the own transmission and is used for error handling if something is physically wrong with the single-wire LIN-bus.

Notes:

- Direct access is enabled only when the transmission is not ongoing (transmission shift register is empty).
- Write a value to the SIOP bit of ESCR register before enabling the transmission (SMR: SOE = 1). This prevents the signal of the unexpected level from being outputted because the SIOP bit retains the previous value.
- During a Read-Modify-Write instruction the SIOP bit returns the actual value of the SOTn pin in the read cycle instead of the value of SINn during a normal read instruction.

20.7.5 Bidirectional Communication Function (Normal Mode)

In operation mode 0 or 2, normal serial bidirectional communication is available. Select operation mode 0 for asynchronous communication and operation mode 2 for synchronous communication.

■ Bidirectional Communication Function

The settings shown in Figure 20.7-10 are required to operate LIN-UART in normal mode (operation mode 0 or 2).

Figure 20.7-10 Settings for LIN-UART Operation Mode 0 and 2

	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SCRn, SMRn	PEN	P	SBL	CL	AD	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 0 →	◎	◎	◎	◎	x	0	◎	◎	0	0	0	◎	0	0	◎	◎
Mode 2 →	▣	▣	▣	+	x	0	◎	◎	1	0	◎	◎	0	0	◎	◎

SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)
Mode 0 →	◎	◎	◎	◎	◎	◎	◎	◎	
Mode 2 →	▣	◎	▣	◎	◎	◎	◎	◎	

ESCRn, ECCRn	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES		LBR	MS	SCDE	SSM		RBI	TBI
Mode 0 →	x	x	x	x	◎	◎	0	0		0	x	x	x	0	◎	◎
Mode 2 →	x	x	x	x	◎	◎	▣	◎		x	◎	◎	◎	0	▣	▣

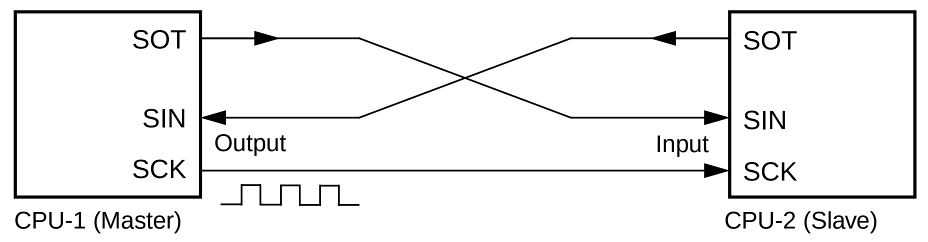
◎: Used bit
x : Unused bit
1 : Set 1
0 : Set 0
▣: Bit used if SSM = 1 (Synchronous start-/stop-bit mode)
+ : Bit automatically set to correct value

n = 0, 1

● Inter-CPU connection

As shown in Figure 20.7-11 , interconnect two CPUs in LIN-UART mode 2

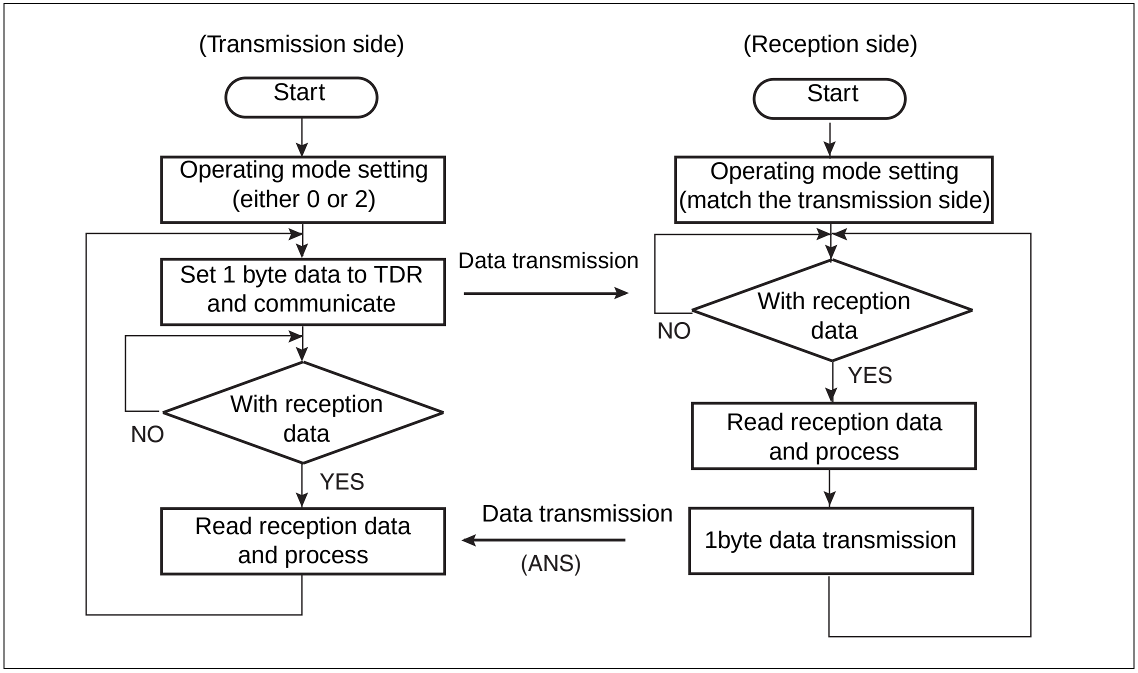
Figure 20.7-11 Connection Example of LIN-UART Mode 2 Bidirectional Communication



● Communication procedure

Communication starts at arbitrary timing from the transmission side when the transmission data is provided. When the transmission data is received at the reception side, ANS (per one byte in example) is returned periodically. Figure 20.7-12 shows an example of the bi-directional communication flowchart.

Figure 20.7-12 Example of Master-slave Communication Flowchart


(Transmission side)
(Reception side)
Start
Start
Operating mode setting (either 0 or 2)
Operating mode setting (match the transmission side)
Set 1 byte data to TDR and communicate
Data transmission
With reception data
NO
YES
With reception data
NO
YES
Read reception data and process
Read reception data and process
Data transmission
(ANS)
1byte data transmission

20.7.6 Master-Slave Communication Function (Multiprocessor Mode)

LIN-UART communication with multiple CPUs connected in master-slave mode is available for both master or slave systems in the operation mode 1.

■ Master-slave Communication Function

The settings shown in Figure 20.7-13 are required to operate LIN-UART in multiprocessor mode (operation mode 1).

Figure 20.7-13 Settings for LIN-UART Operation Mode 1

	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SCRn, SMRn	PEN	P	SBL	CL	AD	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 1 ──►	+	x	◎	◎	◎	0	◎	◎	0	1	0	◎	0	0	◎	◎
SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)							
Mode 1 ──►	x	◎	◎	◎	◎	◎	◎	◎								
ESCRn, ECCRn	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES		LBR	MS	SCDE	SSM		RBI	TBI
Mode 1 ──►	x	x	x	x	◎	◎	0	0		x	x	x	x	0	◎	◎

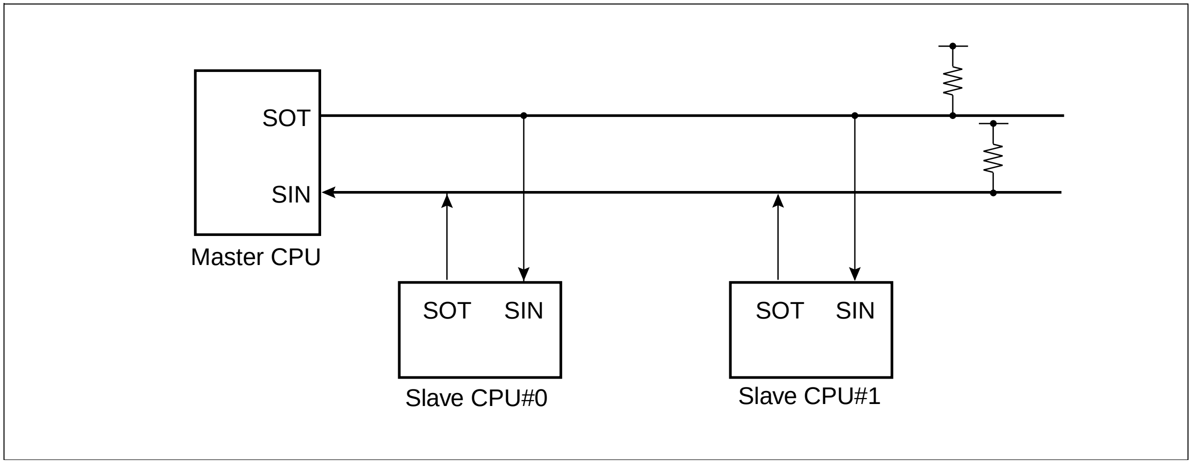
◎ : Used bit
x : Unused bit
1 : Set 1
0 : Set 0
+ : Bit automatically set to correct value

n = 0, 1

● Inter-CPU connection

As shown in Figure 20.7-14 , a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. LIN-UART can be used for the master or slave CPU.

Figure 20.7-14 Connection Example of LIN-UART Master-slave Communication



● Function selection

Select the operation mode and data transfer mode for master-slave communication as shown in Table 20.7-3 .

Table 20.7-3 Selection of the Master-slave Communication Function

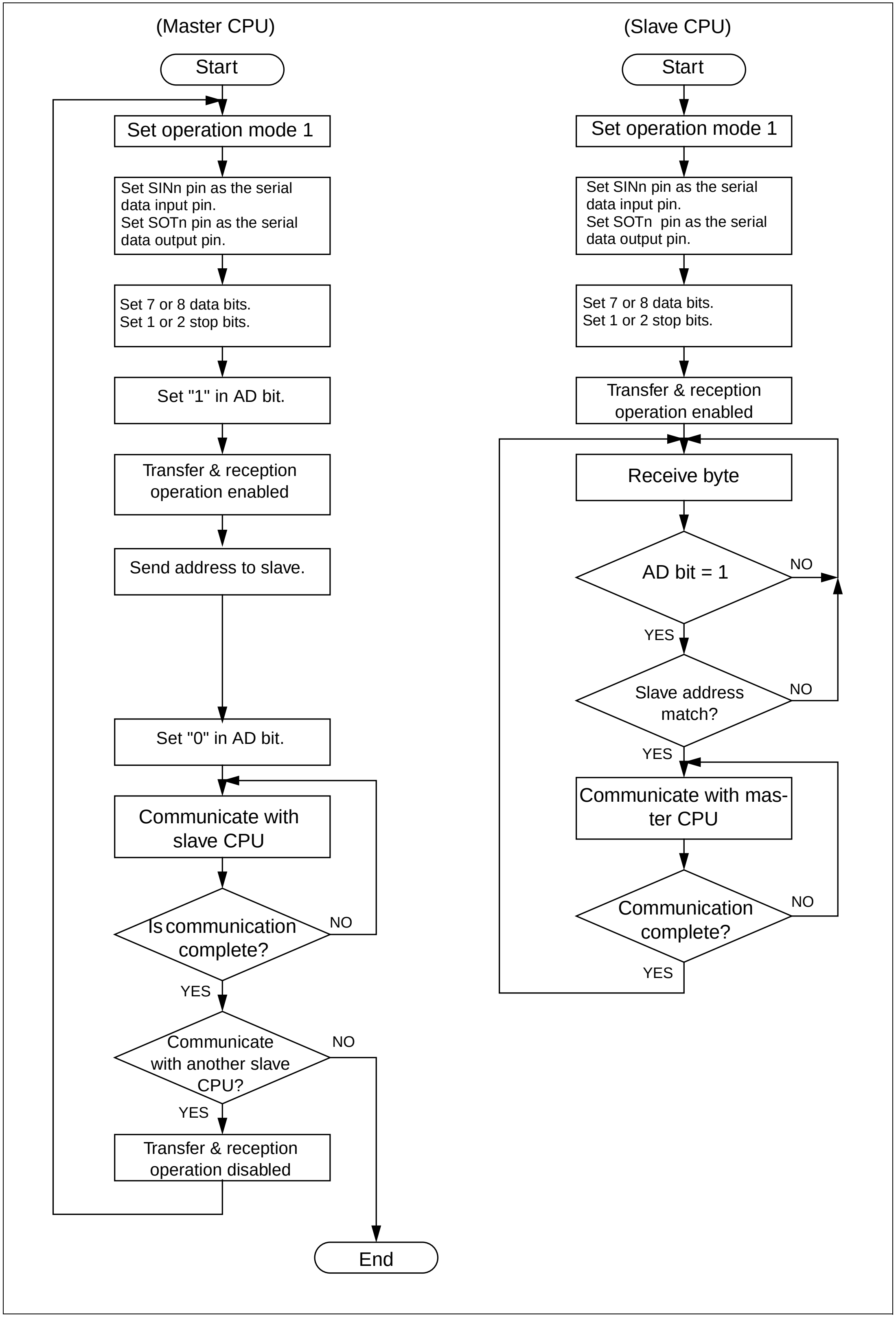
	Operation mode		Data	Parity	Synchronization method	Stop bit	Bit direction
	Master CPU	Slave CPU					
Address transmission and reception	Mode 1 (transmit/ receive AD-bit)	Mode 1 (transmit/ receive AD-bit)	AD="1" + 7- or 8-bit address	None	Asynchronous	1 bit or 2 bits	LSB or MSB first
Data transmission and reception			AD="0" + 7- or 8-bit data				

● Communication procedure

When the master CPU transmits address data, communication starts. The AD bit in the address data is set to 1, and the communication destination slave CPU is selected. Each slave CPU checks the address data using a program. When the address data indicates the address assigned to a slave CPU, the slave CPU communicates with the master CPU.

Figure 20.7-15 shows a flowchart of master-slave communication (multiprocessor mode)

Figure 20.7-15 Master-slave Communication Flowchart


(Master CPU)
Start
Set operation mode 1
Set SINn pin as the serial data input pin.
Set SOTn pin as the serial data output pin.
Set 7 or 8 data bits.
Set 1 or 2 stop bits.
Set "1" in AD bit.
Transfer & reception operation enabled
Send address to slave.
Set "0" in AD bit.
Communicate with slave CPU
Is communication complete?
NO
YES
Communicate with another slave CPU?
NO
YES
Transfer & reception operation disabled
End
(Slave CPU)
Start
Set operation mode 1
Set SINn pin as the serial data input pin.
Set SOTn pin as the serial data output pin.
Set 7 or 8 data bits.
Set 1 or 2 stop bits.
Transfer & reception operation enabled
Receive byte
AD bit = 1
NO
YES
Slave address match?
NO
YES
Communicate with master CPU
Communication complete?
NO
YES

20.7.7 LIN Communication Function

LIN-UART communication with LIN devices is available for both LIN master or LIN slave systems.

■ LIN-master-slave Communication Function

The settings shown in the figure below are required to operate LIN-UART in LIN communication mode (operation mode 3).

Figure 20.7-16 Settings for LIN-UART in Operation Mode 3 (LIN)

SCRn, SMRn	PEN	P	SBL	CL	AD	CRE	RXE	TXE	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Mode 3 →	+	x	+	+	x	0	◎	◎	1	1	0	◎	0	0	◎	◎

SSRn, TDRn/RDRn	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE	Set conversion data (during writing) Retain reception data (during reading)
Mode 3 →	x	◎	◎	◎	◎	+	◎	◎	

ESCRx, ECCRx	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES		LBR	MS	SCDE	SSM		RBI	TBI
Mode 3 →	◎	◎	◎	◎	◎	◎	0	0		◎	x	x	x	0	◎	◎

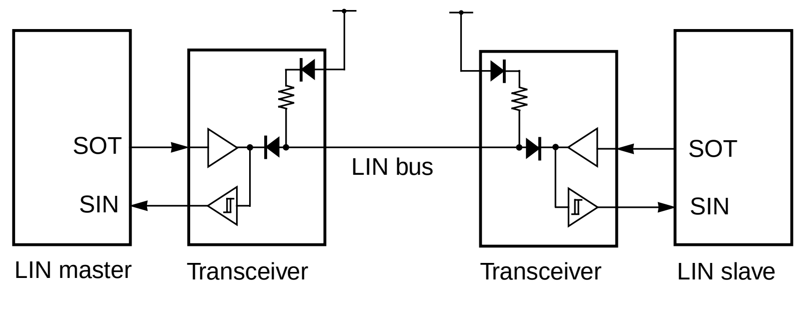
◎: Used bit
x : Unused bit
1 : Set 1
0 : Set 0
+ : Bit automatically set to correct value

n = 0, 1

● LIN device connection

As shown in the Figure below, a communication system of one LIN-Master device and a LIN-Slave device. LIN-UART can operate both as LIN-Master or LIN-Slave.

Figure 20.7-17 Connection Example of a Small LIN-Bus System

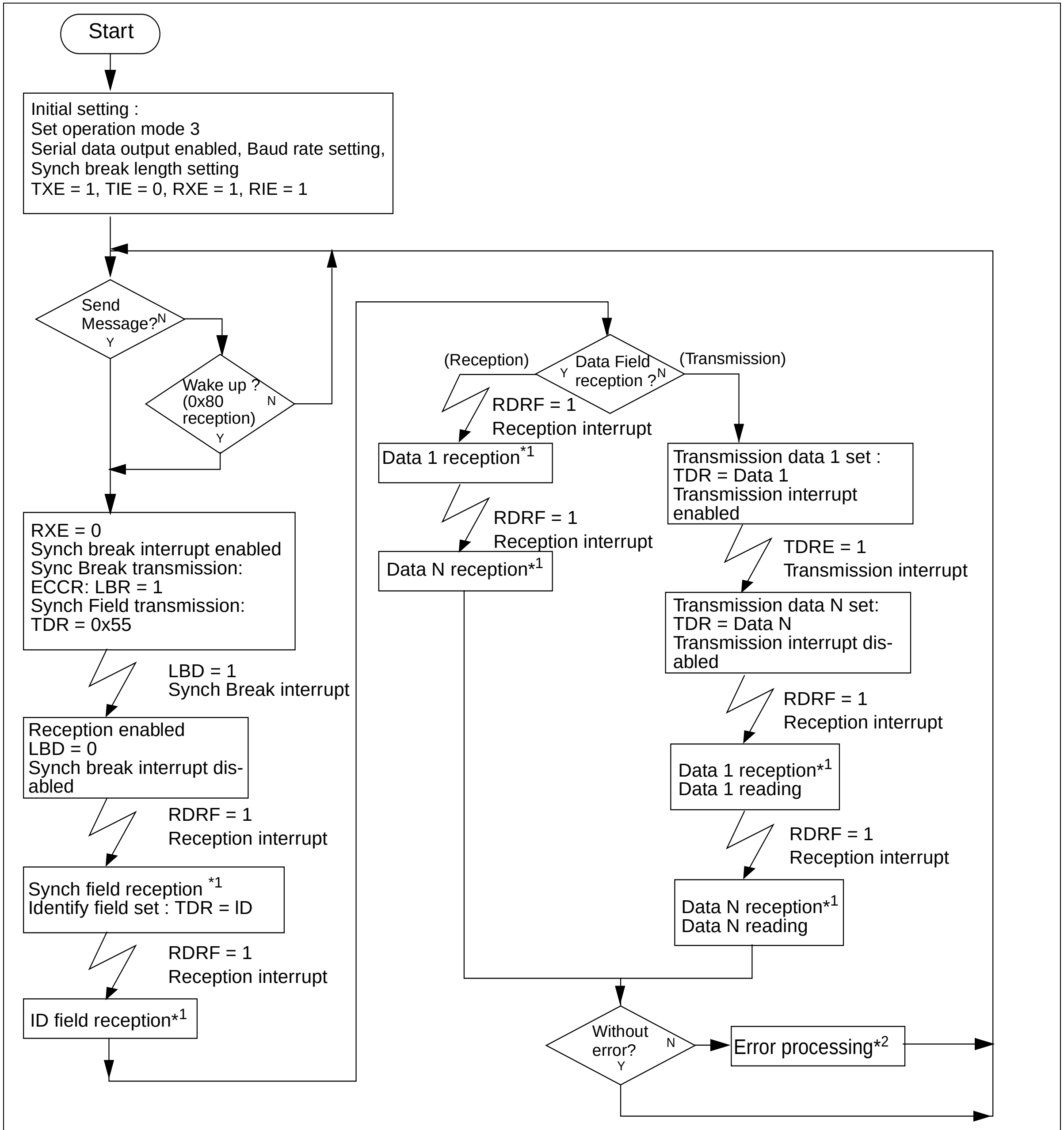


20.7.8 Sample Flowcharts for LIN-UART in LIN communication (Operation Mode 3)

This section contains sample flowcharts for LIN-UART in LIN communication.

■ LIN-UART as LIN Master Device

Figure 20.7-18 LIN-UART LIN Master Flowchart



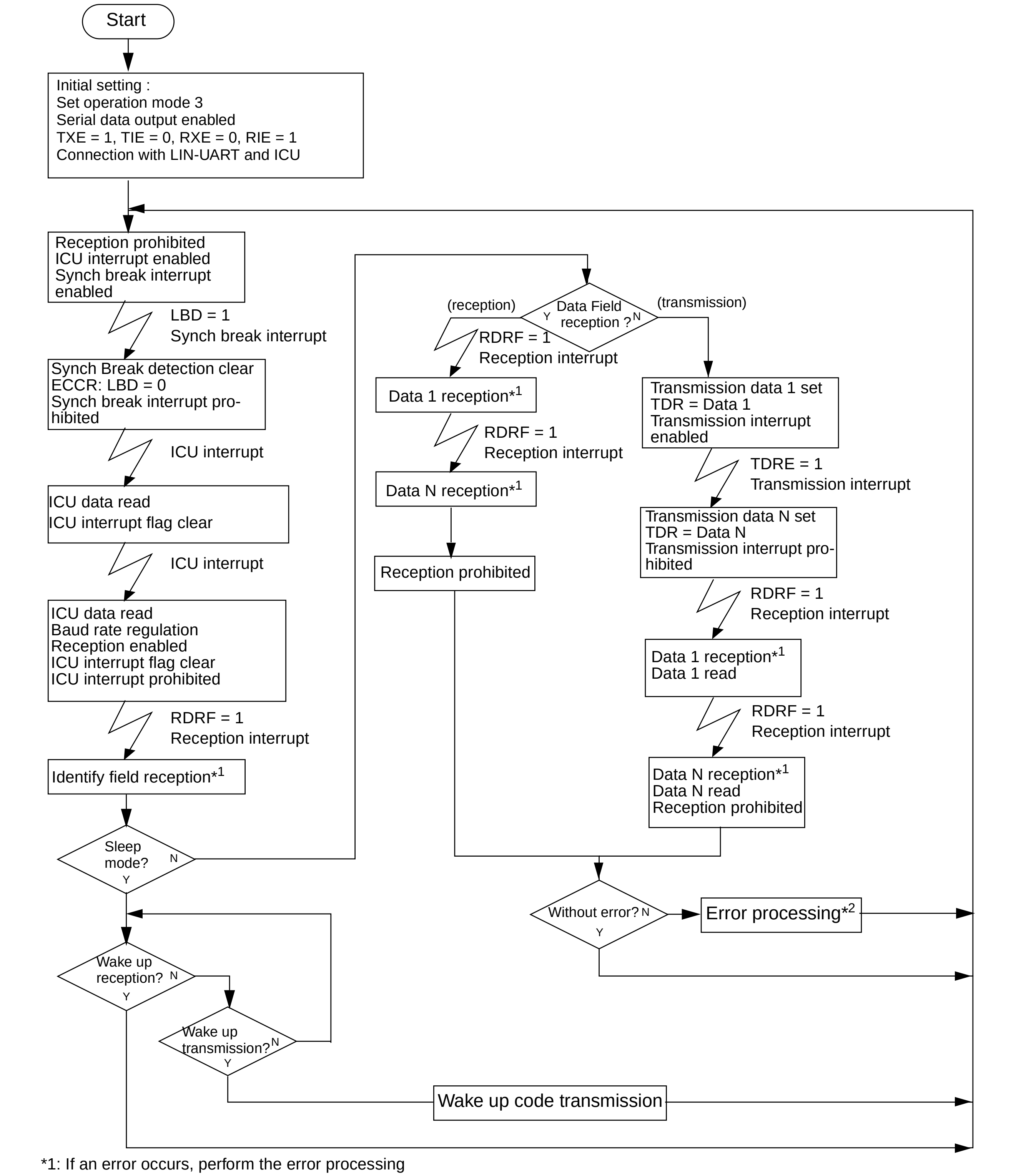
*1: If an error occurs, perform the error processing

*2:
- When fre and ore is "1", write 1 to SCR: CRE bit and clear the error flag.
- When ESCR: LBD bit is "1", execute UART reset.

Note: The error is detected in each processing and take appropriate measure.

■ LIN-UART as LIN slave device

Figure 20.7-19 LIN-UART LIN Slave Flowchart



*1: If an error occurs, perform the error processing

*2: • When fre and ore is "1", write 1 to SCR: CRE bit and clear the error flag.
• When ESCR: LBD bit is "1", execute UART reset.

Note: The error is detected in each processing and take appropriate measure.

20.8 Notes on Using LIN-UART

Notes on using LIN-UART are given below.

■ Notes on Using LIN-UART

● Enabling operations

In LIN-UART, the serial control register (SCR) has TXE (transmission) and RXE (reception) operation enable bits. Both, transmission and reception operations, must be enabled before the communication starts because they have been disabled as the default value (initial value). The operation can also be canceled by disabling these bits.

● Communication mode setting

Set the communication mode while the system is not operating. If the mode is changed during transmission or reception, the transmission or reception is stopped and possible data will be lost.

● Transmission interrupt enabling timing

The default (initial value) of the transmission data empty flag bit (SSR: TDRE) is "1" (no transmission data and transmission data write enable state). A transmission interrupt request is generated as soon as the transmission interrupt request is enabled (SSR: TIE=1). Be sure to set the TIE flag to "1" after setting the transmission data to avoid an immediate interrupt.

● Changing operation settings

It is strongly recommended to reset LIN-UART after changing operation settings. Particularly if (for example) start-/stop-bits added to or removed from the data format.

If settings in the serial mode register (SMR) are desired, it is not useful to set the UPCL bit to 1 at the same time to reset LIN-UART. The correct operation settings are not guaranteed in this case. Thus it is recommended to set the bits of the SMR and then to reset them again plus the UPCL bit.

● Using LIN operation mode 3

The LIN features are available in mode 3 (transmitting, receiving synch break), but using mode 3 sets the UART data format automatically to LIN format (8N1, LSB first). Note that the length of the synch break for transmission is variable but for reception it is fixed 11-bit time.

● LIN slave settings

Set the baud rate before receiving the first LIN synch break for the slave operation. This is needed to detect the minimum of 13-bit time of a LIN synch break surely.

● Software compatibility

Although this LIN-UART is similar to other LIN-UART in other microcontrollers, it is not software compatible to them. The programming models may be the same, but the structure of the registers differ. Furthermore, the setting of the baud rate is now determined by a reload value instead of selecting a predefined value.

● Bus idle function

The bus idle function cannot be used in synchronous mode 2.

● AD bit (serial control register (SCR): address/data type select bit)

Special care has to be taken when using the AD bit (Address-Data-Bit for multiprocessor mode 1) of the Serial Control Register. This bit is both a control and a flag bit, because writing to it sets the AD bit for transmission, whereas reading from it returns the last received AD bit. Internally, the received and the transmitted value are stored in different registers, but in Read-Modify-Write instructions, the transmitted value is read. This can lead to a wrong value in the AD bit, when one of the other bits in the same register is accessed by an instruction of this kind.

Therefore, this bit should be written by the last register access before transmission. Alternatively, using byte wise access and writing the correct values for all bits at once avoids this problem.

● Software reset of LIN-UART

Perform the software reset (SMR: UPCL=1), when the TXE bit of the SCR register is "0".

● LIN Synch detection

In mode 3 (LIN operation), the LBD bit in the ESCR register is set to "1" if the input signal is kept at "0" for more than equal to 11-bit time. Then the LIN-UART waits for the following synch field to be received. If the LIN-UART is set into this state for other reasons than the synch break, it recognizes that synch break is inputted (LBD = 1) and waits for synch field.

In this case, execute the LIN-UART reset (SMR: UPCL = 1).

CHAPTER 21

CAN CONTROLLER

This chapter explains the functions and operations of the CAN controller.

21.1 Features of CAN Controller

The CAN (Controller Area Network) is the standard protocol for serial communication between automobile controllers and is widely used in industrial applications.

■ Features of CAN Controller

- Conforms to CAN Specification Version 2.0 Part A and B

  Supports transmission/reception in standard frame and extended frame formats
- Supports transmitting of data frames by receiving remote frames
- 16 transmitting/receiving message buffers
  - 29-bit ID and 8 bytes data
  - Multi-level message buffer configuration
- Supports full-bit comparison, full-bit mask and partial bit mask filtering

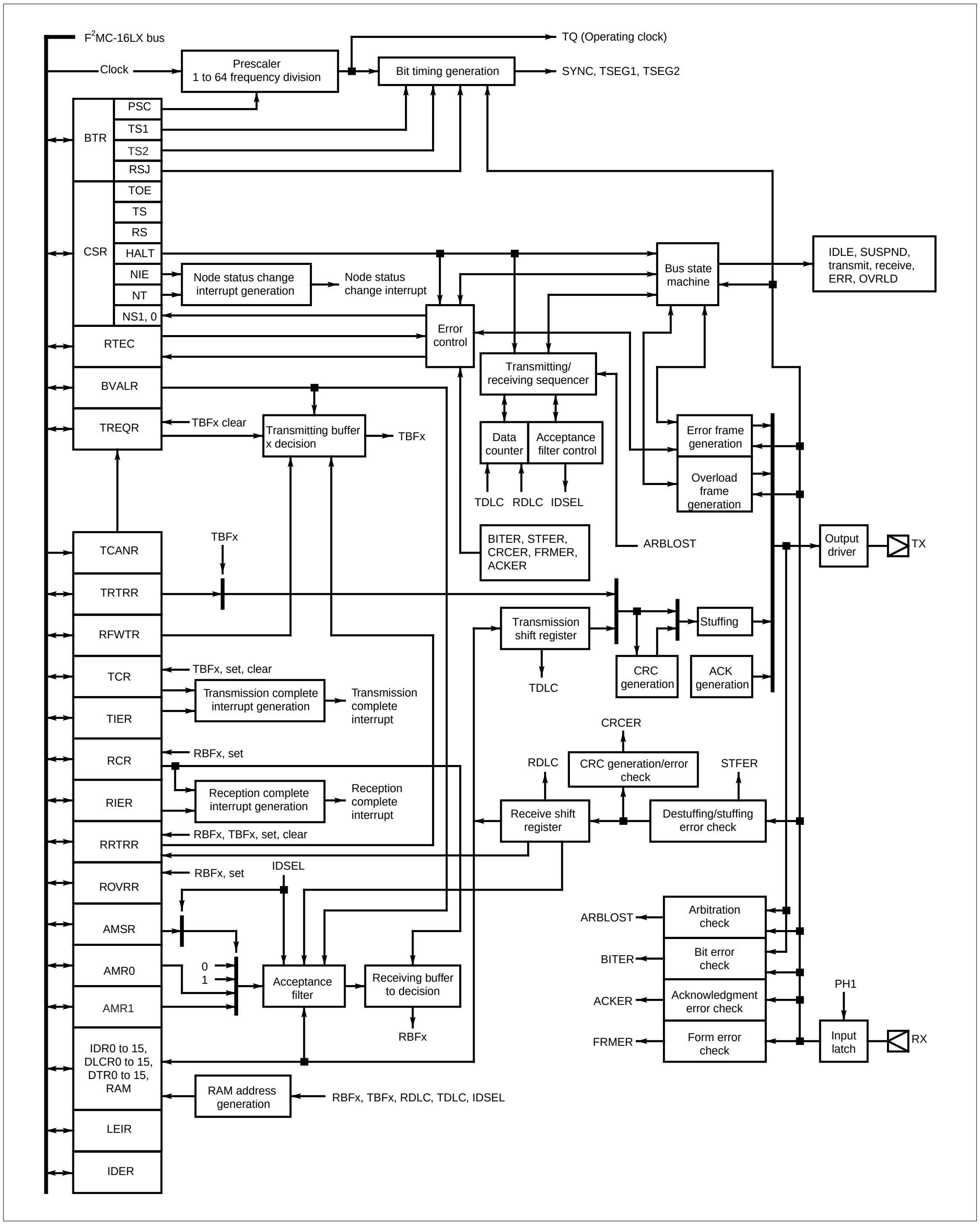
  Two acceptance mask registers in either standard frame format or extended frame formats
- Bit rate programmable from 10 Kbps to 1 Mbps (Minimum 8 MHz machine clock is required at using 1 Mbps.)

21.2 Block Diagram of CAN Controller

Figure 21.2-1 shows a block diagram of the CAN controller.

■ Block Diagram of CAN Controller

Figure 21.2-1 Block Diagram of CAN Controller



F²MC-16LX bus
Clock
Prescaler
1 to 64 frequency division
Bit timing generation
TQ (Operating clock)
SYNC, TSEG1, TSEG2
BTR
PSC
TS1
TS2
RSJ
CSR
TOE
TS
RS
HALT
NIE
NT
NS1, 0
Node status change interrupt generation
Node status change interrupt
Bus state machine
IDLE, SUSPND, transmit, receive, ERR, OVRLD
Error control
RTEC
BVALR
TREQR
TBFx clear
Transmitting buffer x decision
TBFx
Transmitting/ receiving sequencer
Data counter
Acceptance filter control
TDLC
RDLC
IDSEL
Error frame generation
Overload frame generation
BITER, STFER, CRCER, FRMER, ACKER
ARBLOST
Output driver
TX
TCANR
TBFx
TRTRR
RFWTR
Transmission shift register
Stuffing
TDLC
CRC generation
ACK generation
TCR
TBFx, set, clear
Transmission complete interrupt generation
Transmission complete interrupt
TIER
CRCER
RCR
RBFx, set
RDLC
CRC generation/error check
STFER
RIER
Reception complete interrupt generation
Reception complete interrupt
Receive shift register
Destuffing/stuffing error check
RRTRR
RBFx, TBFx, set, clear
ROVRR
RBFx, set
IDSEL
AMSR
ARBLOST
Arbitration check
AMR0
0
1
Acceptance filter
Receiving buffer to decision
BITER
Bit error check
AMR1
ACKER
Acknowledgment error check
PH1
RBFx
FRMER
Form error check
Input latch
RX
IDR0 to 15, DLCR0 to 15, DTR0 to 15, RAM
RAM address generation
RBFx, TBFx, RDLC, TDLC, IDSEL
LEIR
IDER

21.3 List of Overall Control Registers

Following Table lists the register.

■ List of overall Control Registers

Table 21.3-1 List of overall Control Register (1 / 2)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
000080_H	Message buffer valid register	BVALR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
000081_H				
000082_H	Transmit request register	TREQR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
000083_H				
000084_H	Transmit cancel register	TCANR	W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
000085_H				
000086_H	Transmit complete register	TCR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
000087_H				
000088_H	Receive complete register	RCR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
000089_H				
$00008A_H$	Remote request receiving register	RRTRR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$00008B_H$				
$00008C_H$	Receive overrun register	ROVRR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$00008D_H$				
$00008E_H$	Receive interrupt enable register	RIER	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$00008F_H$				
$007D00_H$	Control status register	CSR	R/W, R	0 0 XXX 0 0 0 0 XXXX0 X1
$007D01_H$				
$007D02_H$	Last event indicator register	LEIR	R/W	XXXXXXXX 0 0 0 X0 0 0 0
$007D03_H$				
$007D04_H$	Receive/transmit error counter	RTEC	R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$007D05_H$				
$007D06_H$	Bit timing register	BTR	R/W	X1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$007D07_H$				
$007D08_H$	IDE register	IDER	R/W	XXXXXXXX XXXXXXXX
$007D09_H$				
$007D0A_H$	Transmit RTR register	TRTRR	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$007D0B_H$				

Table 21.3-1 List of overall Control Register (2 / 2)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
007D0C$_H$	Remote frame receive waiting register	RFWTR	R/W	XXXXXXXX XXXXXXXX
007D0D$_H$				
007D0E$_H$	Transmit interrupt enable register	TIER	R/W	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
007D0F$_H$				
007D10$_H$	Acceptance mask select register	AMSR	R/W	XXXXXXXX XXXXXXXX
007D11$_H$				
007D12$_H$				XXXXXXXX XXXXXXXX
007D13$_H$				
007D14$_H$	Acceptance mask register 0	AMR0	R/W	XXXXXXXX XXXXXXXX
007D15$_H$				
007D16$_H$				XXXXXXXX XXXXXXXX
007D17$_H$				
007D18$_H$	Acceptance mask register 1	AMR1	R/W	XXXXXXXX XXXXXXXX
007D19$_H$				
007D1A$_H$				XXXXXXXX XXXXXXXX
007D1B$_H$				

■ List of Message Buffers (ID registers)

Table 21.3-2 List of Message Buffers (ID registers) (1 / 2)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
007C00$_H$ to 007C1F$_H$	General-purpose RAM	–	R/W	XXXXXXXX to XXXXXXXX
007C20$_H$	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX
007C21$_H$				
007C22$_H$				XXXXXXXX XXXXXXXX
007C23$_H$				
007C24$_H$	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX
007C25$_H$				
007C26$_H$				XXXXXXXX XXXXXXXX
007C27$_H$				
007C28$_H$	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX
007C29$_H$				
007C2A$_H$				XXXXXXXX XXXXXXXX
007C2B$_H$				
007C2C$_H$	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX
007C2D$_H$				
007C2E$_H$				XXXXXXXX XXXXXXXX
007C2F$_H$				
007C30$_H$	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX
007C31$_H$				
007C32$_H$				XXXXXXXX XXXXXXXX
007C33$_H$				
007C34$_H$	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX
007C35$_H$				
007C36$_H$				XXXXXXXX XXXXXXXX
007C37$_H$				
007C38$_H$	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX
007C39$_H$				
007C3A$_H$				XXXXXXXX XXXXXXXX
007C3B$_H$				

Table 21.3-2 List of Message Buffers (ID registers) (2 / 2)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
007C3C$_H$	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX
007C3D$_H$				
007C3E$_H$				XXXXXXXX XXXXXXXX
007C3F$_H$				
007C40$_H$	ID register 8	IDR8	R/W	XXXXXXXX XXXXXXXX
007C41$_H$				
007C42$_H$				XXXXXXXX XXXXXXXX
007C43$_H$				
007C44$_H$	ID register 9	IDR9	R/W	XXXXXXXX XXXXXXXX
007C45$_H$				
007C46$_H$				XXXXXXXX XXXXXXXX
007C47$_H$				
007C48$_H$	ID register 10	IDR10	R/W	XXXXXXXX XXXXXXXX
007C49$_H$				
007C4A$_H$				XXXXXXXX XXXXXXXX
007C4B$_H$				
007C4C$_H$	ID register 11	IDR11	R/W	XXXXXXXX XXXXXXXX
007C4D$_H$				
007C4E$_H$				XXXXXXXX XXXXXXXX
007C4F$_H$				
007C50$_H$	ID register 12	IDR12	R/W	XXXXXXXX XXXXXXXX
007C51$_H$				
007C52$_H$				XXXXXXXX XXXXXXXX
007C53$_H$				
007C54$_H$	ID register 13	IDR13	R/W	XXXXXXXX XXXXXXXX
007C55$_H$				
007C56$_H$				XXXXXXXX XXXXXXXX
007C57$_H$				
007C58$_H$	ID register 14	IDR14	R/W	XXXXXXXX XXXXXXXX
007C59$_H$				
007C5A$_H$				XXXXXXXX XXXXXXXX
007C5B$_H$				
007C5C$_H$	ID register 15	IDR15	R/W	XXXXXXXX XXXXXXXX
007C5D$_H$				
007C5E$_H$				XXXXXXXX XXXXXXXX
007C5F$_H$				

■ List of Message Buffers (DLC registers and data registers)

Table 21.3-3 List of Message Buffer (DLC register and data register)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
007C60$_H$	DLC register 0	DLCR0	R/W	XXXXXXXX
007C61$_H$				
007C62$_H$	DLC register 1	DLCR1	R/W	XXXXXXXX
007C63$_H$				
007C64$_H$	DLC register 2	DLCR2	R/W	XXXXXXXX
007C65$_H$				
007C66$_H$	DLC register 3	DLCR3	R/W	XXXXXXXX
007C67$_H$				
007C68$_H$	DLC register 4	DLCR4	R/W	XXXXXXXX
007C69$_H$				
007C6A$_H$	DLC register 5	DLCR5	R/W	XXXXXXXX
007C6B$_H$				
007C6C$_H$	DLC register 6	DLCR6	R/W	XXXXXXXX
007C6D$_H$				
007C6E$_H$	DLC register 7	DLCR7	R/W	XXXXXXXX
007C6F$_H$				
007C70$_H$	DLC register 8	DLCR8	R/W	XXXXXXXX
007C71$_H$				
007C72$_H$	DLC register 9	DLCR9	R/W	XXXXXXXX
007C73$_H$				
007C74$_H$	DLC register 10	DLCR10	R/W	XXXXXXXX
007C75$_H$				
007C76$_H$	DLC register 11	DLCR11	R/W	XXXXXXXX
007C77$_H$				
007C78$_H$	DLC register 12	DLCR12	R/W	XXXXXXXX
007C79$_H$				
007C7A$_H$	DLC register 13	DLCR13	R/W	XXXXXXXX
007C7B$_H$				
007C7C$_H$	DLC register 14	DLCR14	R/W	XXXXXXXX
007C7D$_H$				
007C7E$_H$	DLC register 15	DLCR15	R/W	XXXXXXXX
007C7F$_H$				

■ List of Message Buffer (data register)

Table 21.3-4 List of Message Buffer (data register)

Address	Register	Abbreviation	Access	Initial Value
CAN1				
$007C80_H$ to $007C87_H$	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX to XXXXXXXX
$007C88_H$ to $007C8F_H$	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX to XXXXXXXX
$007C90_H$ to $007C97_H$	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX to XXXXXXXX
$007C98_H$ to $007C9F_H$	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX to XXXXXXXX
$007CA0_H$ to $007CA7_H$	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX to XXXXXXXX
$007CA8_H$ to $007CAF_H$	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX to XXXXXXXX
$007CB0_H$ to $007CB7_H$	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX to XXXXXXXX
$007CB8_H$ to $007CBF_H$	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX to XXXXXXXX
$007CC0_H$ to $007CC7_H$	Data register 8 (8 bytes)	DTR8	R/W	XXXXXXXX to XXXXXXXX
$007CC8_H$ to $007CCF_H$	Data register 9 (8 bytes)	DTR9	R/W	XXXXXXXX to XXXXXXXX
$007CD0_H$ to $007CD7_H$	Data register 10 (8 bytes)	DTR10	R/W	XXXXXXXX to XXXXXXXX
$007CD8_H$ to $007CDF_H$	Data register 11 (8 bytes)	DTR11	R/W	XXXXXXXX to XXXXXXXX
$007CE0_H$ to $007CE7_H$	Data register 12 (8 bytes)	DTR12	R/W	XXXXXXXX to XXXXXXXX
$007CE8_H$ to $007CEF_H$	Data register 13 (8 bytes)	DTR13	R/W	XXXXXXXX to XXXXXXXX
$007CF0_H$ to $007CF7_H$	Data register 14 (8 bytes)	DTR14	R/W	XXXXXXXX to XXXXXXXX
$007CF8_H$ to $007CFF_H$	Data register 15 (8 bytes)	DTR15	R/W	XXXXXXXX to XXXXXXXX

21.4 Classifying CAN Controller Registers

There are 3 types of CAN controller registers;

- **Overall control registers**
- **Message buffer control registers**
- **Message buffers**

■ Overall Control Registers

The overall control registers are the following 4 registers;

- Control status register (CSR)
- Last event indicator register (LEIR)
- Receive and transmit error counter (RTEC)
- Bit timing register (BTR)

■ Message Buffer Control Registers

The message buffer control registers are the following 14 registers;

- Message buffer valid register (BVALR)
- IDE register (IDER)
- Transmission request register (TREQR)
- Transmission RTR register (TRTRR)
- Remote frame receiving wait register (RFWTR)
- Transmission cancel register (TCANR)
- Transmission complete register (TCR)
- Transmission interrupt enable register (TIER)
- Reception complete register (RCR)
- Remote request receiving register (RRTRR)
- Receive overrun register (ROVRR)
- Reception interrupt enable register (RIER)
- Acceptance mask select register (AMSR)
- Acceptance mask registers 0 and 1 (AMR0 and AMR1)

■ Message Buffers

The message buffers are the following 3 registers;

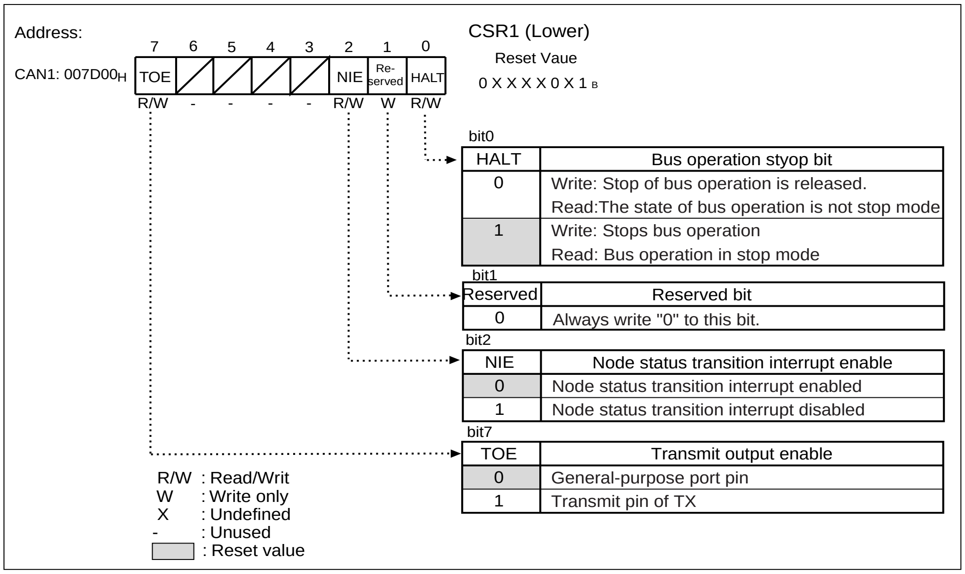
- ID register x (x = 0 to 15) (IDRx)
- DLC register x (x = 0 to 15) (DLCRx)
- Data register x (x = 0 to 15) (DTRx)

21.4.1 Configuration of Control Status Register (CSR)

This register indicates bus operation, node status, transmit output enable and transmit/ receive status. The lower 8-bit with the control status register (CSR) is prohibited from executing any bit manipulation instructions (Read-Modify-Write instructions). Only in the case of HALT bits unchanged (initialization of the macro instructions etc.), there is no problem even if any bit manipulation instructions is used.

■ Control Status Register (CSR) (Lower)

Figure 21.4-1 Configuration of the Control Status Register (lower byte)



HALT	Bus operation styop bit
0	Write: Stop of bus operation is released. Read:The state of bus operation is not stop mode
1	Write: Stops bus operation Read: Bus operation in stop mode

Reserved	Reserved bit
0	Always write "0" to this bit.

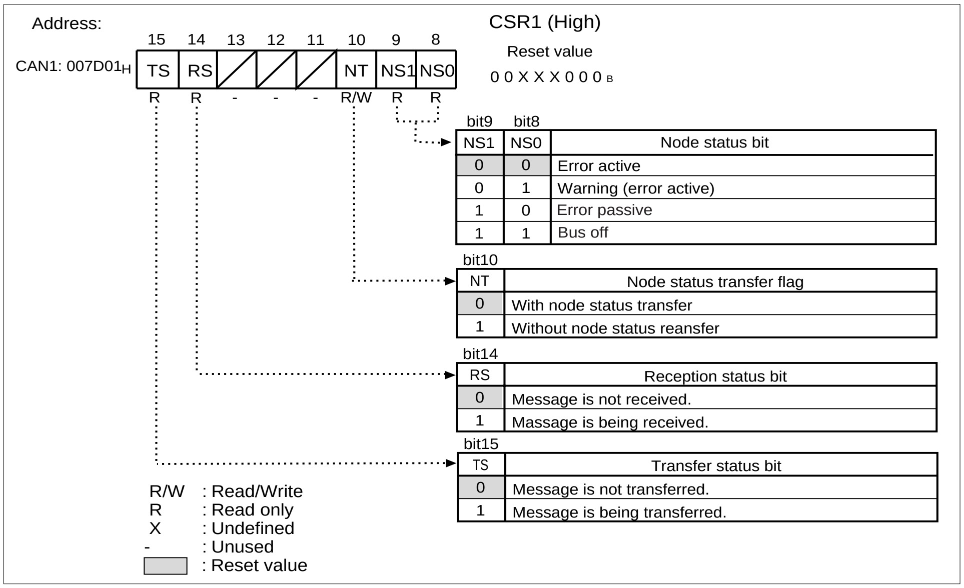
NIE	Node status transition interrupt enable
0	Node status transition interrupt enabled
1	Node status transition interrupt disabled

TOE	Transmit output enable
0	General-purpose port pin
1	Transmit pin of TX

R/W	: Read/Writ
W	: Write only
X	: Undefined
-	: Unused
▒	: Reset value

■ Control Status Register (CSR) (upper)

Figure 21.4-2 Configuration of the Control Status Register (upper byte)



NS1	NS0	Node status bit
0	0	Error active
0	1	Warning (error active)
1	0	Error passive
1	1	Bus off

NT	Node status transfer flag
0	With node status transfer
1	Without node status reansfer

RS	Reception status bit
0	Massage is not received.
1	Massage is being received.

TS	Transfer status bit
0	Message is not transferred.
1	Message is being transferred.

R/W	: Read/Write
R	: Read only
X	: Undefined
-	: Unused
▒	: Reset value

21.4.2 Function of Control Status Register (CSR)

The operating status of the register's each bit is confirmed by following;

- **Setting "0" or "1"**
- **Function control by writing**
- **Read**

■ Control Status Register (CSR-lower)

Table 21.4-1 Function of Each Bit of the Control Status Register (CSR:L)

Bit Name		Function
bit7	TOE: Transmit output enable bit	This bit switches from a general-purpose I/O port to a transmit pin TX. **When setting to 0:** Functions as general-purpose I/O port. **When setting to 1:** Functions as transmit pin TX.
bit6 to bit3	Undefined bits	**When reading:** Value is undefined. **When writing:** No effect
bit2	NIE: Node status transition interrupt output enable bit	This bit controls a node status transition interrupt generation (CSR: NT = 1) when a node status is transferred. **When setting to 0:** Interrupt generation is disabled. **When setting to 1:** Interrupt generation is enabled.
bit1	Reserved: Reserved bit	This bit is always set to 0. **When reading:** The value is always 0.

bit0	HALT:
	Bus operation halt bit
		This bit controls the bus halt. The halt state of the bus can be checked by reading this bit.
		Writing to this bit:
		0: Cancels bus halt
		1: Halt bus
		Reading from this bit:
		0: Bus operation not in stop state
		1: Bus operation in stop state
		Note: After ensuring that 1 is written to this bit, write 0 to this bit if the node status is Bus Off.
		Example program:
		switch (IO_CANCT1.CSR.bit.NS)
		{
		case 0 : /* error active */
		break;
		case 1 : /* warning */
		break;
		case 2 : /* error passive */
		break;
		default : /* bus off */
		for (i=0; (i <= 500) || (IO_CANCT1.CSR.bit.HALT == 0); i++);
		IO_CANCT1.CSR.word = 0x0084; /* HALT = 0 */
		break;
		}
		*: The variable "i" is used for fail-safe.
		For detail information, see "21.4.4 Notes on Using Bus Operation Stop Bit (HALT = 1)".

■ Control status register (CSR-upper)

Table 21.4-2 Function of Each Bit of the Control Status Register (CSR:H)

Bit Name		Function
bit15	TS: Transmit status bit	This bit indicates whether a message is being transmitted. **At read:** 0: Message not being transmitted 1: Message being transmitted This bit is set 0 even while error and overload frames are transmitted.
bit14	RS: Receive status bit	This bit indicates whether a message is being received. **At read:** 0: Message not being received 1: Message being received • While a message is on the bus, this bit becomes 1. Therefore, this bit is also 1 while a message is being transmitted. This bit does not necessarily indicates whether a receiving message passes through the acceptance filter. • As a result, when this bit is 0, it implies that the bus operation is stopped (HALT = 1); the bus is in the intermission/bus idle or a error/overload frame is on the bus.
bit13 to bit11	Undefined bits	**When reading:** The value is undefined. **When writing:** No effect
bit10	NT: Node status transition flag	When the node status changes from increment transition or off bus into error active, this bit is set to "1". The condition that this bit is set to "1" is as follows. At this time, the interruption is generated for the node status interruption permission bit (NIE) = "1". **1)** Error active ("00_B") → Warning ("01_B") **2)** Warning ("01_B") → Error passive ("10_B") **3)** Error passive ("10_B") → Bus off ("11_B") **4)** Bus off ("11_B") → Error active ("00_B") Note: In parentheses, the value of NS1 and the NS0 bit is indicated. **At Write:** "0": Cleared "1": Not possible to set (No effect) **At read by the instruction of the read-modify-write type:** Always read "1".
bit9 bit8	NS1, NS0: Node status bits	These bits indicate the current node status. For detail information, see "21.4.3 Correspondence between Node Status Bit and Node Status".

21.4.3 Correspondence between Node Status Bit and Node Status

Node status bit shows the node status by two bits (NS1 and NS0).

■ Correspondence between Node Status Bit and Node Status

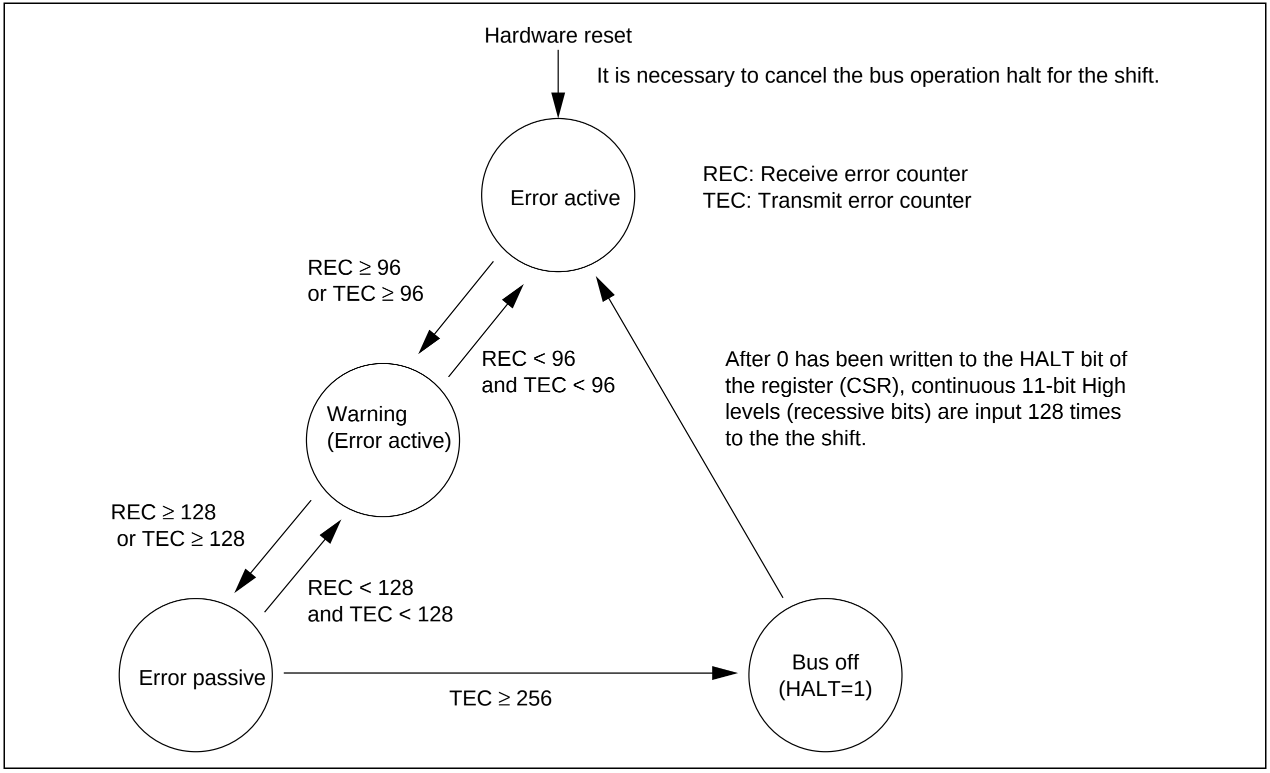
Table 21.4-3 Correspondence between NS1 and NS0 and Node Status

NS1	NS0	Node status
0	0	Error active
0	1	Warning (error active)
1	0	Error passive
1	1	Bus off

Note:

Warning (error active) is included in the error active in CAN Specification 2.0B for the node status, however, indicates that the transmit error counter or receive error counter has exceeded 96. The node status change diagram is shown in Figure 21.4-3 .

Figure 21.4-3 Node Status Transition Diagram


Hardware reset
It is necessary to cancel the bus operation halt for the shift.
Error active
REC: Receive error counter
TEC: Transmit error counter
REC ≥ 96
or TEC ≥ 96
REC < 96
and TEC < 96
After 0 has been written to the HALT bit of the register (CSR), continuous 11-bit High levels (recessive bits) are input 128 times to the the shift.
Warning
(Error active)
REC ≥ 128
or TEC ≥ 128
REC < 128
and TEC < 128
Error passive
Bus off
(HALT=1)
TEC ≥ 256

21.4.4 Notes on Using Bus Operation Stop Bit (HALT = 1)

The bus operation stop bit is set by writing to the bit, hardware reset and the node status. The stop operation of the bus operation is different according to the state of the message buffer.

■ Conditions for Setting Bus Operation Stop (HALT=1)

There are 3 conditions for setting bus operation stop (HALT = 1):

- After hardware reset
- When node status changed to bus off
- By writing 1 to HALT

Notes:

- The bus operation should be stopped by writing 1 to HALT before the F^{2}MC-16LX is changed in low-power consumption mode (stop mode and timebase timer mode). If transmission is in progress when 1 is written to HALT, the bus operation is stopped (HALT = 1) after transmission is terminated. If reception is in progress when 1 is written to HALT, the bus operation is stopped immediately (HALT = 1). If received messages are being stored in the message buffer (x), stop the bus operation (HALT = 1) after storing the messages.
- To check whether the bus operation has stopped, always read the HALT bit.

■ Conditions for Canceling Bus Operation Stop (HALT = 0)

The condition for canceling the bus operation if halt is writing 0 to HALT.

Notes:

- Canceling the bus operation stop after hardware reset or by writing 1 to HALT as above conditions is performed after 0 is written to HALT and continuous 11-bit High levels (recessive bits) have been input to the receive input pin (RX) (HALT = 0).
- Canceling the bus operation stop when the node status is changed to bus off as above conditions is performed after 0 is written to HALT and continuous 11-bit High levels (recessive bits) have been input 128 times to the receive input pin (RX) (HALT = 0). Then, the values of both transmit and receive error counters reach 0 and the node status is changed to error active.
- When write 0 to HALT during the node status is Bus Off, ensure that 1 is written to this bit.

■ State during Bus Operation Stop (HALT = 1)

- The bus does not perform any operation, such as transmission and reception.
- The transmit output pin (TX) outputs a High level (recessive bit).
- The values of other registers and error counters are not changed.

Note:

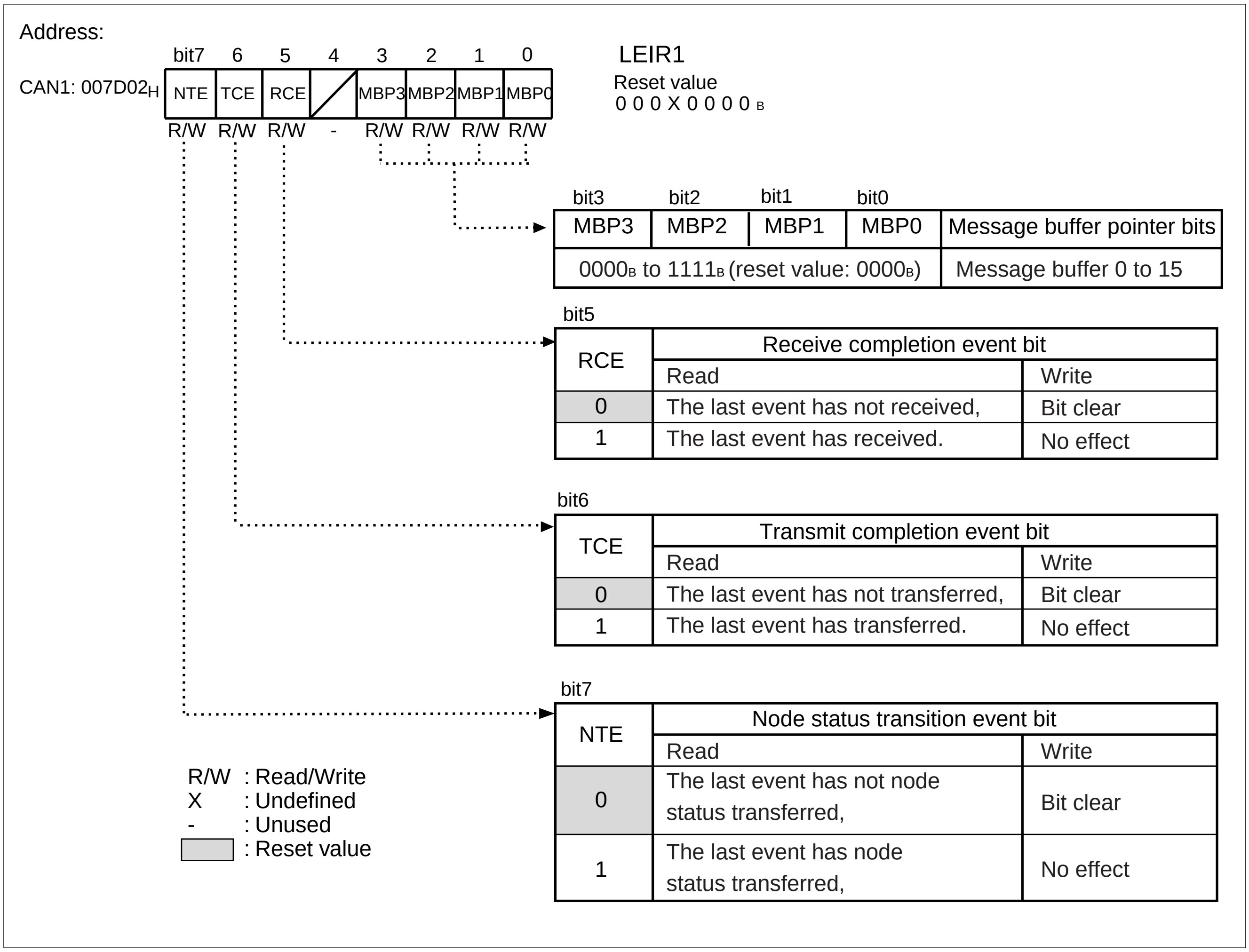
The bit timing register (BTR) should be set during bus operation stop (HALT = 1).

21.4.5 Last Event Indicator Register (LEIR)

This register indicates the last event.
The NTE, TCE, and RCE bits are exclusive. When the corresponding bit of the last event is set to 1, other bits are set to 0.

■ Last Event Indicator Register (LEIR)

Figure 21.4-4 Configuration of the Last Event Indicator Register (LEIR)



Address:

LEIR1

	bit7	6	5	4	3	2	1	0
CAN1: 007D02H	NTE	TCE	RCE		MBP3	MBP2	MBP1	MBP0
	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W

Reset value
0 0 0 X 0 0 0 0 B

bit3	bit2	bit1	bit0	
MBP3	MBP2	MBP1	MBP0	Message buffer pointer bits
0000B to 1111B (reset value: 0000B)				Message buffer 0 to 15

bit5

RCE	Receive completion event bit	
	Read	Write
0	The last event has not received,	Bit clear
1	The last event has received.	No effect

bit6

TCE	Transmit completion event bit	
	Read	Write
0	The last event has not transferred,	Bit clear
1	The last event has transferred.	No effect

bit7

NTE	Node status transition event bit	
	Read	Write
0	The last event has not node status transferred,	Bit clear
1	The last event has node status transferred,	No effect

R/W : Read/Write
X : Undefined
- : Unused
▨ : Reset value

■ Last Event Indicator Register (LEIR)

Table 21.4-4 Function of Each Bit of the Last Event Indicator Register (LEIR) (1 / 2)

Bit Name		Function
bit7	NTE: Node status transition event bit	When this bit is 1, node status transition is the last event. This bit is set to 1 after set either of bit of the control status register to "1" (CSR:NTx=1). • This setting is not related to the setting of NIE bit of the control status register (CSR). At Write: "0": Cleared "1": No effect At read by the instruction of the read-modify-write type: Always read "1".
bit6	TCE: Transmit completion event bit	When this bit is 1, it indicates that transmit completion is the last event. This bit is set to 1 after set either of bit of the transmit completion register to "1" (TCR:TCx=1). • This setting is not related to the setting of the transmit interrupt enable register (TIER). • When this bit is "1", MBP3 to MBP0 bits show the message buffer number (x) to complete the transmission of the message in the last event. At Write: "0": Cleared "1": No effect At read by the instruction of the read-modify-write type: Always read "1".
bit5	RCE: Receive completion event bit	When this bit is 1, it indicates that receive completion is the last event. This bit is set to 1 after set either of bit of the receive complete register to "1" (RCR:RCx=1). • This setting is not related to the setting of the receive interrupt enable register (RIER). • When this bit is "1", MBP3 to MBP0 bits show the message buffer number (x) to complete the transmission of the message in the last event. At Write: "0": Cleared "1": No effect At read by the instruction of the read-modify-write type: Always read "1".
bit4	Undefined bit	**When reading:** The value is undefined. **When writing:** No effect

Table 21.4-4 Function of Each Bit of the Last Event Indicator Register (LEIR) (2 / 2)

Bit Name		Function
bit3 to bit0	MBP3 to MBP0: Message buffer pointer bits	When TCE bit or RCE bit is "1", these bits show the message buffer number (x) to generating of corresponding the last event. If the NTE bit is set to 1, these bits have no meaning. At Write: "0": Cleared "1": No effect At read by the instruction of the read-modify-write type: Always read "1". Note: If LEIR is accessed within an CAN interrupt handler, the event causing the interrupt is not necessarily the same as indicated by LEIR. In the time from interrupt request to the LEIR access by the interrupt handler there may occur other CAN events.

21.4.6 Receive and Transmit Error Counters (RTEC)

The receive and transmit error counters indicate the counts for transmission errors and reception errors defined in the CAN specifications. These registers can only be read.

■ Register Configuration

Figure 21.4-5 Configuration of the Receive and Transmit Error Counters

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RTEC1(Upper)
CAN1:	007D05H	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	Reset value 0 0 0 0 0 0 0 0 B
		R	R	R	R	R	R	R	R	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RTEC1(Lower)
CAN1:	007D04H	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	Reset value 0 0 0 0 0 0 0 0 B
		R	R	R	R	R	R	R	R	

R : Read only

■ Register Function

Table 21.4-5 Function of Each Bit of the Receive and Transmit Error Counters (RTEC)

Bit Name		Function
bit15 to bit8	TEC7 to TEC0: Transmit error counter bits	These are transmit error counters. TEC7 to TEC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Bus Off is indicated for the node status (NS1 and NS0 of control status register CSR = 11).
bit7 to bit0	REC7 to REC0: Receive error counter bits	These are receive error counters. REC7 to REC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Error Passive is indicated for the node status (NS1 and NS0 of control status register CSR = 10).

21.4.7 Bit Timing Register (BTR)

Bit timing register (BTR) sets the prescaler and bit timing setting.

■ Register Configuration

Figure 21.4-6 Configuration of the Bit Timing Register (BTR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	BTR1(Upper)
CAN1:	007D07H	–	TS2.2	TS2.1	TS2.0	TS1.3	TS1.2	TS1.1	TS1.0	Reset value X1 1 1 1 1 1 1 B
		–	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	BTR1(Lower)
CAN1:	007D06H	RSJ1	RSJ0	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0	Reset value 11 1 1 1 1 1 1 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write
X : Undefined
– : Unused

■ Register Function

Table 21.4-6 Function of Each Bit of the Bit Timing Register (BTR)

Bit Name		Function
bit14 to bit12	TS2.2 to TS2.0: Time segment 2 setting bits 2 to 0	These bits define the number of the time quanta (TQ’s) for the time segment 2 (TSEG2). The time segment 2 is equal to the phase buffer segment 2 (PHASE_SEG2) in the CAN specification.
bit11 to bit8	TS1.3 to TS1.0: Time segment 1 setting bits 3 to 0	These bits define the number of the time quanta (TQ’s) for the time segment 1 (TSEG1). The time segment 1 is equal to the propagation segment (PROP_SEG) + phase buffer segment 1 (PHASE_SEG1) in the CAN specification.
bit7 bit6	RSJ1, RSJ0: Resynchronization jump width setting bits 1, 0	These bits define the number of the time quanta (TQ’s) for the resynchronization jump width.
bit5 to bit0	PSC5 to PSC0: Prescaler setting bits 5 to 0	These bits define the time quanta (TQ) of the CAN controller. (see below for details.)

Note: Please set (CSR: HALT=1) to bit timing register (BTR) after stopping the bus operation. Please release the bus operation stop by writing "0" in the HALT bit of the control status register after the setting of bit timing register (BTR) is ended.

21.4.8 Prescaler Setting by Bit Timing Register (BTR)

The setting of bit timing register (BTR) corresponds to the bit time of prescaler in the CAN specification and the CAN controller segment.

■ Prescaler Settings

The bit time segments defined in the CAN specification, and the CAN controller are shown in Figure 21.4-7 and Figure 21.4-8 respectively.

Figure 21.4-7 Bit Time Segment in CAN Specification

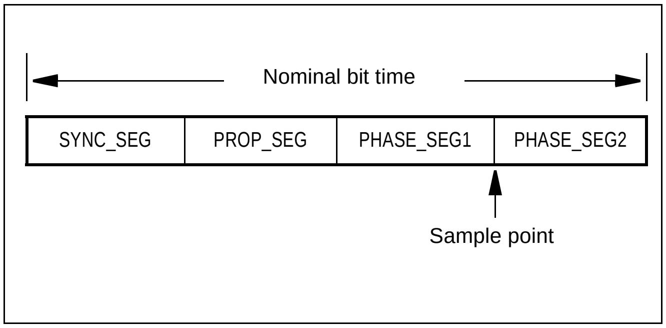
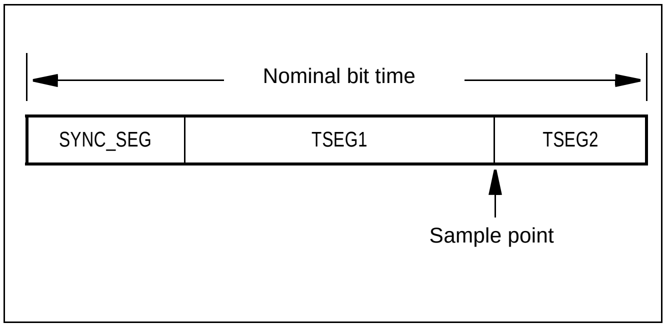


Figure 21.4-8 Bit Time Segment in CAN Controller



The relationship between PSC = PSC5 to PSC0, TSI = TS1.3 to TS1.0, TS2 = TS2.2 to TS2.0, and RSJ = RSJ1, RSJ0

$$\begin{aligned} TQ &= (PSC + 1) \text{ x } CLK \\ BT &= SYNC\_SEG + TSEG1 + TSEG2 \\ &= (1 + (TS1 + 1) + (TS2 + 1)) \text{ x } TQ \\ &= (3 + TS1 + TS2) \text{ x } TQ \\ RSJW &= (RSJ + 1) \text{ x } TQ \end{aligned}$$

RSJ1 and RSJ0 is shown below.

CLK: input clock (CLK)

TQ: time quanta

BT: bit time

SYNC_SEG: synchronous segment

TSEG1 and TSEG2: time segment 1 and 2

resynchronization jump width [(RSJ1 and RSJ0) +1] frequency division

For correct operation, the following conditions should be met.

$$\begin{aligned} &\text{For } 1 \leqq PSC \leqq 63: \\ &\quad TSEG1 \geqq 2TQ \\ &\quad TSEG1 \geqq RSJW \\ &\quad TSEG2 \geqq 2TQ \\ &\quad TSEG2 \geqq RSJW \\ &\text{For } PSC = 0: \\ &\quad TSEG1 \geqq 5TQ \\ &\quad TSEG2 \geqq 2TQ \\ &\quad TSEG2 \geqq RSJW \end{aligned}$$

In order to meet the bit timing requirements defined in the CAN specification, additions have to be met, e.g. the propagation delay has to be considered.

21.4.9 Message Buffer Valid Register (BVALR)

Message buffer valid register (BVALR) stores the validity of the message buffers or displays their state.

■ Register Configuration

Figure 21.4-9 Configuration of the Message Buffer Valid Register (BVALR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	BVRLR1(Upper)
CAN1:	000081H	BVAL15	BVAL14	BVAL13	BVAL12	BVAL11	BVAL10	BVAL9	BVAL8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	BVRLR1(Lower)
CAN1:	000080H	BVAL7	BVAL6	BVAL5	BVAL4	BVAL3	BVAL2	BVAL1	BVAL0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

0: Message buffer (x) invalid

1: Message buffer (x) valid

If the message buffer (x) is set to invalid, it will not transmit or receive messages.

If the buffer is set to invalid during transmission operating, it becomes invalid (BVALx = 0) after the transmission is completed or terminated by an error.

If the buffer is set to invalid during reception operating, it immediately becomes invalid (BVALx = 0). If received messages are stored in a message buffer (x), the message buffer (x) is invalid after storing the messages.

Notes:

- x indicates a message buffer number (x = 0 to 15).
- When invaliding a message buffer (x) by writing 0 to a bit (BVALx), execution of a bit manipulation instruction is prohibited until the bit is set to 0.
- To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in "21.13 Precautions when Using CAN Controller".

21.4.10 IDE Register (IDER)

This register stores the frame format used by the message buffers (x) during transmission/reception.

■ Register Configuration

Figure 21.4-10 Configuration of the IDE Register (IDER)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	IDER1(upper)
CAN1:	007D09$_H$	IDE15	IDE14	IDE13	IDE12	IDE11	IDE10	IDE9	IDE8	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	IDER1(Lower)
CAN1:	007D08$_H$	IDE7	IDE6	IDE5	IDE4	IDE3	IDE2	IDE1	IDE0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write
X : Undefined

■ Register Function

0: The standard frame format (ID11 bit) is used for the message buffer (x).

1: The extended frame format (ID29 bit) is used for the message buffer (x).

Notes:

- This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) = 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.
- To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in "21.13 Precautions when Using CAN Controller".

21.4.11 Transmission Request Register (TREQR)

Transmission request register (TREQR) stores transmission requests to the message buffers (x) or displays their state.

■ Register Configuration

Figure 21.4-11 Configuration of the Transmission Request Register (TREQR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TREQR1(Upper)
CAN1:	000083H	TREQ15	TREQ14	TREQ13	TREQ12	TREQ11	TREQ10	TREQ9	TREQ8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TREQR1(Lower)
CAN1:	000082H	TREQ7	TREQ6	TREQ5	TREQ4	TREQ3	TREQ2	TREQ1	TREQ0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

When 1 is written to TREQx, transmission to the message buffer (x) starts.

If RFWTx of the remote frame receiving wait register (RFWTR) *1 is 0, transmission starts immediately. However, if RFWTx = 1, transmission starts after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR)*1 becomes 1). Transmission starts *2 immediately even when RFWTx = 1, if RRTRx is already 1 when 1 is written to TREQx.

*1: For RFWTR and TRTRR, see "21.4.12 Transmission RTR Register (TRTRR)" and "21.4.13 Remote Frame Receiving Wait Register (RFWTR)".

*2: For cancellation of transmission, see "21.4.14 Transmission Cancel Register (TCANR)" and "21.4.15 Transmission Complete Register (TCR)".

Writing 0 to TREQx is ignored.

0 is read when a Read Modify Write instruction is performed.

If clearing (to 0) at completion of the transmit operation and setting by writing 1 are concurrent, clearing is preferred.

If 1 is written to more than 1 bit, transmission is performed, starting with the lower-numbered message buffer (x).

TREQx is 1 while transmission is pending, and becomes 0 when transmission is completed or canceled.

21.4.12 Transmission RTR Register (TRTRR)

This register stores the RTR (Remote Transmission Request) bits for the message buffers (x).

■ Register Configuration

Figure 21.4-12 Configuration of the Transmission RTR Register (TRTRR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TRTRR1(Upper)
CAN1:	007D0BH	TRTR15	TRTR14	TRTR13	TRTR12	TRTR11	TRTR10	TRTR9	TRTR8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TRTRR1(Lower)
CAN1:	007D0AH	TRTR7	TRTR6	TRTR5	TRTR4	TRTR3	TRTR2	TRTR1	TRTR0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

0: Transmit data frame.

1: Transmit remote frame.

21.4.13 Remote Frame Receiving Wait Register (RFWTR)

Remote frame receiving wait register (RFWTR) sets the conditions for starting transmission when a request for data frame transmission is set (TREQx of the transmission request register (TREQR) is 1 and TRTRx of the transmitting RTR register (TRTRR) is 0).

■ Register Configuration

Figure 21.4-13 Configuration of the Remote Frame Receiving Wait Register (RFWTR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RFWTR1(Upper)
CAN1:	007D0D$_H$	RFWT15	RFWT14	RFWT13	RFWT12	RFWT11	RFWT10	RFWT9	RFWT8	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RFWTR1(Lower)
CAN1:	007D0C$_H$	RFWT7	RFWT6	RFWT5	RFWT4	RFWT3	RFWT2	RFWT1	RFWT0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write
X : Undefined

■ Register Function

0: Transmission starts immediately

1: Transmission starts after waiting until remote frame received (RRTRx of remote request receiving register (RRTRR) becomes 1)

Notes:

- Transmission starts immediately if RRTRx is already 1 when a request for transmission is set.
- For remote frame transmission, do not set RFWTx to 1.

21.4.14 Transmission Cancel Register (TCANR)

This register cancels a pending request for transmission to the message buffer (x).

■ Register Configuration

Figure 21.4-14 Configuration of the Transmission Cancel Register (TCANR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TCANR1(Upper)
CAN1:	000085H	TCAN15	TCAN14	TCAN13	TCAN12	TCAN11	TCAN10	TCAN9	TCAN8	Reset value 0 0 0 0 0 0 0 0 B
		W	W	W	W	W	W	W	W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TCANR1(Lower)
CAN1:	000084H	TCAN7	TCAN6	TCAN5	TCAN4	TCAN3	TCAN2	TCAN1	TCAN0	Reset value 0 0 0 0 0 0 0 0 B
		W	W	W	W	W	W	W	W	

W : Write only

■ Register Function

When 1 is written to TCANx, this register cancels a pending request for transmission to the message buffer (x). At completion of cancellation, TREQx of the transmission request register (TREQR) becomes 0.

Writing 0 to TCANx is ignored.

This is a write-only register and its read value is always 0.

21.4.15 Transmission Complete Register (TCR)

At completion of transmission by the message buffer (x), the corresponding TCx becomes 1.
If TIEx of the transmission complete interrupt enable register (TIER) is 1, an interrupt occurs.

■ Register Configuration

Figure 21.4-15 Configuration of the Transmission Complete Register (TCR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TCR1(Upper)
CAN1:	000087H	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TCR1(Lower)
CAN1:	000086H	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write
X : Undefined

■ Register Function

● Conditions for TCx = 0

- Write 0 to TCx.
- Write 1 to TREQx of the transmission request register (TREQR).

After the completion of transmission, write 0 to TCx to set it to 0. Writing 1 to TCx is ignored.

1 is read when a Read Modify Write instruction is performed.

Note:

If setting to 1 by completion of the transmit operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

21.4.16 Transmission Interrupt Enable Register (TIER)

This register enables or disables the transmission interrupt by the message buffer (x). The transmission interrupt is generated at transmission completion (when TCx of the transmission complete register (TCR) is 1).

■ Register Configuration

Figure 21.4-16 Configuration of the Transmission Interrupt Enable Register (TIER)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	TIER1(Upper)
CAN1:	007D0FH	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	TIER1(Lower)
CAN1:	007D0EH	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

0: Transmission interrupt disabled.

1: Transmission interrupt enabled.

21.4.17 Reception Complete Register (RCR)

At completion of storing received message in the message buffer (x), RCx becomes 1. If RIEx of the reception complete interrupt enable register (RIER) is 1, an interrupt occurs.

■ Register Configuration

Figure 21.4-17 Configuration of the Reception Complete Register (RCR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RCR1(Upper)
CAN1:	000089H	RC15	RC14	RC13	RC12	RC11	RC10	RC9	RC8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RCR1(Lower)
CAN1:	000088H	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

● Conditions for RCx = 0

Write 0 to RCx.

After completion of handling received message, write 0 to RCx to set it to 0. Writing 1 to RCx is ignored.

1 is read when a Read Modify Write instruction is performed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

21.4.18 Remote Request Receiving Register (RRTRR)

After a remote frame is stored in the message buffer (x), RRTRx becomes 1 (at the same time as RCx setting to 1).

■ Register Configuration

Figure 21.4-18 Configuration of the Remote Request Receiving Register (RRTRR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RRTRR1(Upper)
CAN1:	00008B$_H$	RRTR15	RRTR14	RRTR13	RRTR12	RRTR11	RRTR10	RRTR9	RRTR8	Reset value 0 0 0 0 0 0 0 0$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RRTRR1(Lower)
CAN1:	00008A$_H$	RRTR7	RRTR6	RRTR5	RRTR4	RRTR3	RRTR2	RRTR1	RRTR0	Reset value 0 0 0 0 0 0 0 0$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

● Conditions for RRTRx = 0

- Write 0 to RRTRx.
- After a received data frame is stored in the message buffer (x) (at the same time as RCx setting to 1).
- Transmission by the message buffer (x) is completed (TCx of the transmission complete register (TCR) is 1).

Writing 1 to RRTRx is ignored.

1 is read when a Read Modify Write instruction is performed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

21.4.19 Receive Overrun Register (ROVRR)

If RCx of the reception complete register (RCR) is 1 when completing storing of a received message in the message buffer (x), ROVRx becomes 1, indicating that reception has overrun.

■ Register Configuration

Figure 21.4-19 Configuration of the Receive overrun Register (ROVRR)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	ROVRR1(Upper)
CAN1:	00008DH	RVOR15	RVOR14	RVOR13	RVOR12	RVOR11	RVOR10	RVOR9	RVOR8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	ROVRR1(Lower)
CAN1:	00008CH	RVOR7	RVOR6	RVOR5	RVOR4	RVOR3	RVOR2	RVOR1	RVOR0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

Writing 0 to ROVRx results in ROVRx = 0. Writing 1 to ROVRx is ignored. After checking that reception has overrun, write 0 to ROVRx to set it to 0.

1 is read when a Read-Modify-Write instruction is performed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

21.4.20 Reception Interrupt Enable Register (RIER)

Reception interrupt enable register (RIER) enables or disables the reception interrupt by the message buffer (x).
The reception interrupt is generated at reception completion (when RCx of the reception completion register (RCR) is 1).

■ Register Configuration

Figure 21.4-20 Configuration of the Reception Interrupt Enable Register (RIER)

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	RIER1(Upper)
CAN1:	00008FH	RIE15	RIE14	RIE13	RIE12	RIE11	RIE10	RIE9	RIE8	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	RIER1(Lower)
CAN1:	00008EH	RIE7	RIE6	RIE5	RIE4	RIE3	RIE2	RIE1	RIE0	Reset value 0 0 0 0 0 0 0 0 B
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write

■ Register Function

0: Reception interrupt disabled.

1: Reception interrupt enabled.

21.4.21 Acceptance Mask Select Register (AMSR)

This register selects masks (acceptance mask) for comparison between the received message ID's and the message buffer ID.

■ Register Configuration

Figure 21.4-21 Configuration of the Acceptance Mask Select Register (AMSR)

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMSR1(Byte0)
CAN1:	007D10H	AMS3.1	AMS3.0	AMS2.1	AMS2.0	AMS1.1	AMS1.0	AMS0.1	AMS0.0	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMSR1(Byte1)
CAN1:	007D11H	AMS7.1	AMS7.0	AMS6.1	AMS6.0	AMS5.1	AMS5.0	AMS4.1	AMS4.0	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMSR1(Byte2)
CAN1:	007D12H	AMS11.1	AMS11.0	AMS10.1	AMS10.0	AMS9.1	AMS9.0	AMS8.1	AMS8.0	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMSR1(Byte3)
CAN1:	007D13H	AMS15.1	AMS15.0	AMS14.1	AMS14.0	AMS13.1	AMS13.0	AMS12.1	AMS12.0	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write
X : Undefined

■ Register Function

Table 21.4-7 Selection of Acceptance Mask

AMSx.1	AMSx.0	Acceptance Mask
0	0	Full-bit comparison
0	1	Full-bit mask
1	0	Acceptance mask register 0 (AMR0)
1	1	Acceptance mask register 1 (AMR1)

Notes:

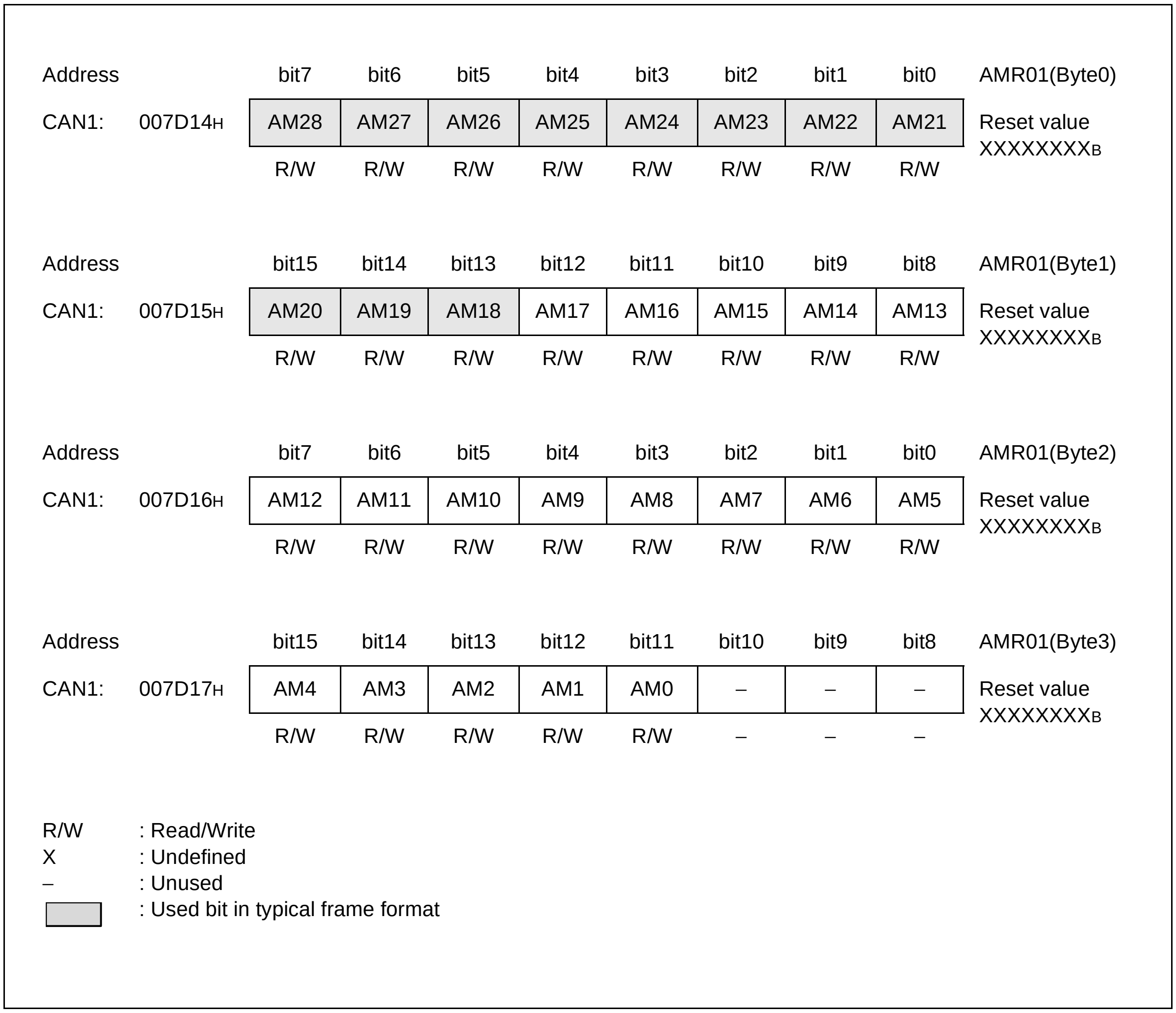
- AMSx.1 and AMSx.0 should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.
- To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in "21.13 Precautions when Using CAN Controller".

21.4.22 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)

There are two acceptance mask registers, which are available either in the standard frame format or extended frame format.
AM28 to AM18 (11 bits) are used for acceptance masks in the standard frame format and AM28 to AM0 (29 bits) are used for acceptance masks in the extended format.

■ Register Configuration

Figure 21.4-22 Configuration of the Acceptance Mask Register 0 (AMR0)



Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR01(Byte0)
CAN1:	007D14$_H$	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR01(Byte1)
CAN1:	007D15$_H$	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR01(Byte2)
CAN1:	007D16$_H$	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR01(Byte3)
CAN1:	007D17$_H$	AM4	AM3	AM2	AM1	AM0	–	–	–	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	–	–	–	

R/W : Read/Write
X : Undefined
– : Unused
▨ : Used bit in typical frame format

Figure 21.4-23 Configuration of the Acceptance Mask Register 1 (AMR1)

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR11(Byte0)
CAN1:	007D18H	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR11(Byte1)
CAN1:	007D19H	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	AMR11(Byte2)
CAN1:	007D1AH	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	AMR11(Byte3)
CAN1:	007D1BH	AM4	AM3	AM2	AM1	AM0	–	–	–	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	–	–	–	

R/W	: Read/Write
X	: Undefined
–	: Unused
▨	: Used bit in typical frame format

■ Register Function

● 0: Compare

Compare the bit (be set to "0") of the acceptance code (ID register IDRx for comparing with the received message ID) corresponding to this bit with the bit of the received message ID. If there is no match, no message is received.

● 1: Mask

Mask the bit of the acceptance code ID register (IDRx) corresponding to this bit. No comparison is made with the bit of the received message ID.

Notes:

- AMR0 and AMR1 should be set when all the message buffers (x) selecting AMR0 and AMR1 are invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffers are valid (BVALx = 1) may cause unnecessary received messages to be stored.
- To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in "21.13 Precautions when Using CAN Controller".

21.4.23 Message Buffers

There are 16 message buffers. Message buffer x (x = 0 to 15) consists of an ID register (IDRx), DLC register (DLCRx), and data register (DTRx).

■ Message Buffers

● Register Configuration

- ID register x (x = 0 to 15) (IDRx)

  This register is a ID register of the message buffer. This register memorizes receipt code setting, transmission message ID setting, and reception ID.

- DLC register x (x = 0 to 15) (DLCRx)

  This register stores the DLC of the message buffer. This register sets the data length of the message when a data frame and a remote frame are transmitted and the data length of the message when a data frame or a remote frame is received.

- Data register x (x = 0 to 15) (DTRx)

  This register is a data register of the message buffer. This register memorizes the setting or the reception message data of the transmission message data.

● The message buffer (x) is used both for transmission and reception.

● The lower-numbered message buffers are assigned higher priority.

- At transmission, when a request for transmission is made to more than 1 message buffer, transmission is performed, starting with the lowest-numbered message buffer (See "21.5 Transmission of CAN Controller").
- At reception, when the received message ID passes through the acceptance filter (mechanism for comparing the acceptance-masked ID of received message and message buffer) of more than 1 message buffer, the received message is stored in the lowest-numbered message buffer (See "21.6 Reception of CAN Controller").

● Message buffer that can be used as multi level message buffer

When the same receipt filter is set in 1 or more message buffers, the message buffer can be used as a multi level message buffer.

As a result, the reserve to the reception time is given. (See "21.10 Procedure for Reception by Message Buffer (x)").

Notes:

- A write operation to message buffers and general-purpose RAM areas should be performed in words to even addresses only. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.
- When the BVALx bit of the message buffer valid register (BVALR) is 0 (Invalid), the message buffers x (IDRx, DLCRx, and DTRx) can be used as general-purpose RAM.

  During the receive/transmit operation of the CAN controller, the CAN Controller write/read to/from the message buffers. If the CPU tries to write/read to/from the message buffers in this period, the CPU has to wait a maximum time of 64 machine cycles.

  This is also true for the general-purpose RAM (Address 007A00$_H$ to 007A1F$_H$, 007C00$_H$ to 007C1F$_H$, 007E00$_H$ to 007E1F$_H$).

21.4.24 ID Register x (x = 0 to 15) (IDRx)

This register is the ID register for message buffer (x).

■ Register Configuration

Figure 21.4-24 Configuration of the ID Registers (IDRx)

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	IDRx1(Byte0)
CAN1:	007C20H + 4 × x	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	IDRx1(Byte1)
CAN1:	007C21H + 4 × x	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	IDRx1(Byte2)
CAN1:	007C22H + 4 × x	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	IDRx1(Byte3)
CAN1:	007C23H + 4 × x	ID4	ID3	ID2	ID1	ID0	–	–	–	Reset value XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	–	–	–	

x = 0, ..., 15

R/W	: Read/Write
X	: Undefined
–	: Unused
▨	: Used bit in typical frame format

■ Register Function

When using the message buffer (x) in the standard frame format (IDEx of the IDE register (IDER) = 0), use 11 bits of ID28 to ID18. When using the buffer in the extended frame format (IDEx = 1), use 29 bits of ID28 to ID0.

ID28 to ID0 have the following functions;

- Set acceptance code (ID for comparing with the received message ID).
- Set transmitted message ID.

Note: In the standard frame format, setting 1s to all bits of ID28 to ID22 is prohibited).

- Store the received message ID.

Note: All received message ID bits are stored (even if bits are masked). In the standard frame format, ID17 to ID0 stores image of old message left in the receive shift register.

Notes:

- A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.
- This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.
- To invalidate the message buffer (by setting the BVALR: BVAL bit to 0) while the CAN controller is operating for CAN communication (the read value of the CSR: HALT bit is 0 and the CAN controller is operating for CAN bus communication to enable transmission and reception), follow the procedure in "21.13 Precautions when Using CAN Controller".

21.4.25 DLC Register x (x = 0 to 15) (DLCRx)

This register is the DLC register for message buffer (x).

■ Register Configuration

Figure 21.4-25 Configuration of the DLC Registers (DLCRx)

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DLCR1x(Lower)
CAN1:	$007C60_H + 2 \times x$	–	–	–	–	DLC3	DLC2	DLC1	DLC0	Reset value $XXXXXXXX_B$
		–	–	–	–	R/W	R/W	R/W	R/W	
										x = 0, ..., 15

R/W : Read/Write
X : Undefined
– : Unused

■ Register Function

● Transmission

- Set the data length (byte count) of a transmitted message when a data frame is transmitted (TRTRx of the transmitting RTR register (TRTRR) is 0).
- Set the data length (byte count) of a requested message when a remote frame is transmitted (TRTRx = 1).

Note:

Setting other than 0000_B to 1000_B (0 to 8 bytes) is prohibited.

● Reception

- Store the data length (byte count) of a received message when a data frame is received (RRTRx of the remote frame request receiving register (RRTRR) is 0).
- Store the data length (byte count) of a requested message when a remote frame is received (RRTRx = 1).

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

21.4.26 Data Register x (x = 0 to 15) (DTRx)

This register is the data register for message buffer (x).
This register is used only in transmitting and receiving a data frame but not in transmitting and receiving a remote frame.

■ Register Configuration

Figure 21.4-26 Configuration of the Data Registers (DTRx)

Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx1(Byte0)
CAN1:	007C80$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx1(Byte1)
CAN1:	007C81$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx1(Byte2)
CAN1:	007C82$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx1(Byte3)
CAN1:	007C83$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx1(Byte4)
CAN1:	007C84$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx1(Byte5)
CAN1:	007C85$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	DTRx1(Byte6)
CAN1:	007C86$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	DTRx1(Byte7)
CAN1:	007C87$_H$ + 8 × x	D7	D6	D5	D4	D3	D2	D1	D0	Reset value XXXXXXXX$_B$
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

x = 0, ..., 15

R/W : Read/Write
X : Undefined
– : Unused

■ Register Function

- Sets transmitted message data (any of 0 to 8 bytes).

  Data is transmitted in the order of BYTE0, BYTE1, ..., BYTE7, starting with the MSB.

- Stores received message data.

  Data is stored in the order of BYTE0, BYTE1, ..., BYTE7, starting with the MSB.

  Even if the received message data is less than 8 bytes, the remaining bytes of the data register (DTRx), to which data are stored, are undefined.

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

21.5 Transmission of CAN Controller

When 1 is written to TREQx of the transmission request register (TREQR), transmission by the message buffer (x) starts. At this time, TREQx becomes 1 and TCx of the transmission complete register (TCR) becomes 0.

■ Starting Transmission of CAN Controller

If RFWTx of the remote frame receiving wait register (RFWTR) is 0, transmission starts immediately. If RFWTx is 1, transmission starts after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR) becomes 1).

If a request for transmission is made to more than 1 message buffer (more than one TREQx is 1), transmission is performed, starting with the lowest-numbered message buffer.

Message transmission to the CAN bus (by the transmit output pin TX) starts when the bus is idle.

If TRTRx of the transmission RTR register (TRTRR) is 0, a data frame is transmitted. If TRTRx is 1, a remote frame is transmitted.

If the message buffer competes with other CAN controllers on the CAN bus for transmission and arbitration fails, or if an error occurs during transmission, the message buffer waits until the bus is idle and repeats retransmission until it is successful.

■ Canceling Transmission Request from CAN Controller

● Canceling by transmission cancel register (TCANR)

A transmission request for message buffer (x) having not executed transmission during transmission pending can be canceled by writing 1 to TCANx of the transmission cancel register (TCANR). At completion of cancellation, TREQx becomes 0.

● Canceling by storing received message

The message buffer (x) having not executed transmission despite transmission request also performs reception.

If the message buffer (x) has not executed transmission despite a request for transmission of a data frame (TRTRx = 0 or TREQx = 1), the transmission request is canceled after storing received data frames passing through the acceptance filter (TREQx = 0).

Note:

A transmission request is not canceled by storing remote frames (TREQx = 1 remains unchanged).

If the message buffer (x) has not executed transmission despite a request for transmission of a remote frame (TRTRx = 1 or TREQx = 1), the transmission request is canceled after storing received remote frames passing through the acceptance filter (TREQx = 0).

Note:

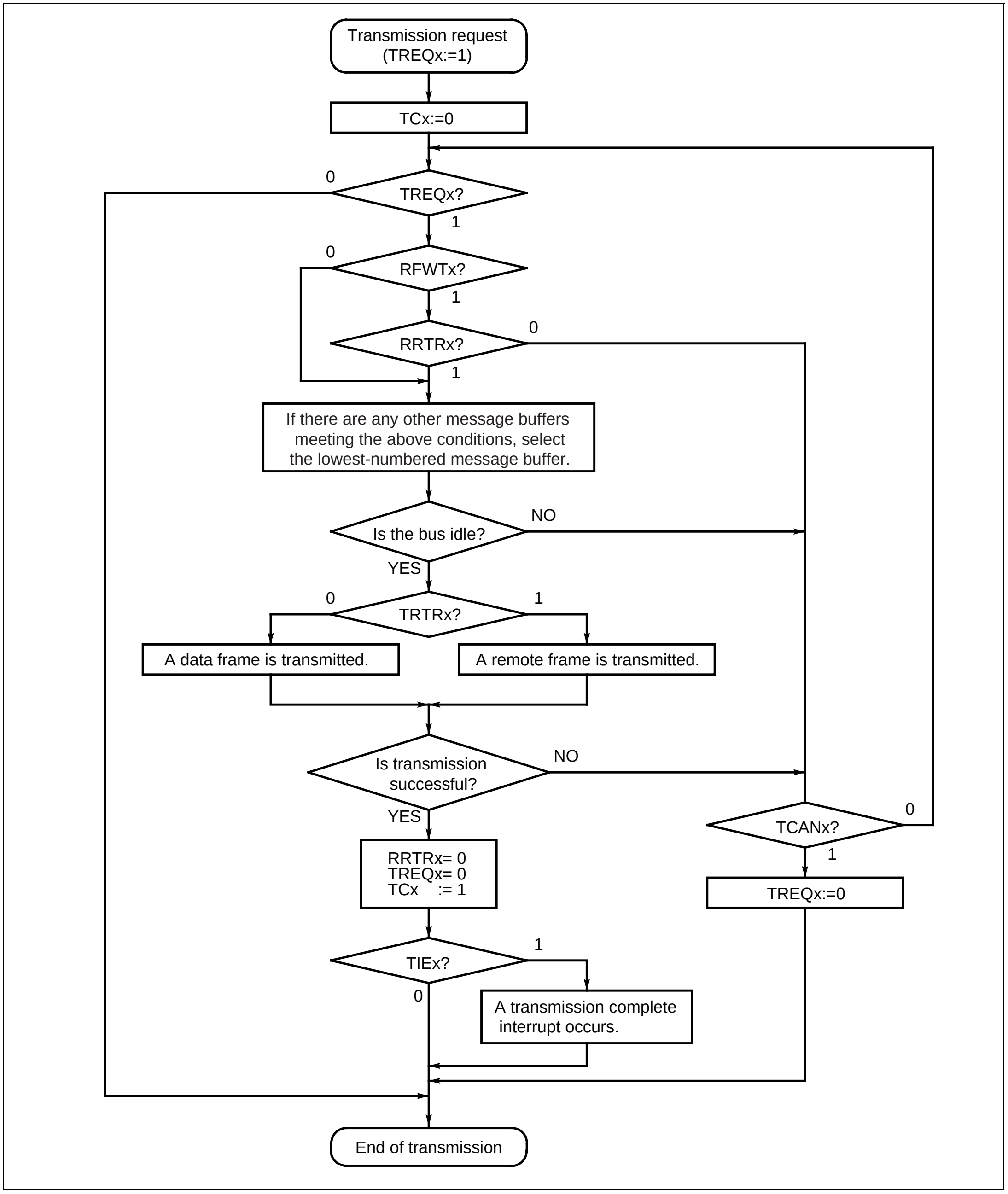
The transmission request is canceled by storing either data frames or remote frames.

■ Completing Transmission of CAN Controller

When transmission is successful, RRTRx becomes 0, TREQx becomes 0, and TCx of the transmission complete register (TCR) becomes 1. If the transmission complete interrupt is enabled (TIEx of the transmission complete interrupt enable register (TIER) is 1), an interrupt occurs.

■ Transmission Flowchart of CAN Controller

Figure 21.5-1 Transmission Flowchart of the CAN Controller


Transmission request
(TREQx:=1)
TCx:=0
TREQx?
0
1
RFWTx?
0
1
RRTRx?
0
1
If there are any other message buffers meeting the above conditions, select the lowest-numbered message buffer.
Is the bus idle?
NO
YES
TRTRx?
0
1
A data frame is transmitted.
A remote frame is transmitted.
Is transmission successful?
NO
YES
TCANx?
0
1
RRTRx= 0
TREQx= 0
TCx := 1
TREQx:=0
TIEx?
1
0
A transmission complete interrupt occurs.
End of transmission

21.6 Reception of CAN Controller

Reception starts when the start of data frame or remote frame (SOF) is detected on the CAN bus.

■ Acceptance Filtering

The received message in the standard frame format is compared with the message buffer (x) set in the standard frame format (IDEx of the IDE register (IDER) is 0). The received message in the extended frame format is compared with the message buffer (x) set (IDEx is 1) in the extended frame format.

If all the bits set to Compare by the acceptance mask agree after comparison between the received message ID and acceptance code (ID register (IDRx) for comparing with the received message ID), the received message passes to the acceptance filter of the message buffer (x).

■ Storing Received Message

When the receive operation is successful, received messages are stored in a message buffer x including IDs passed through the acceptance filter.

When receiving data frames, received messages are stored in the ID register (IDRx), DLC register (DLCRx), and data register (DTRx).

Even if received message data is less than 8 bytes, some data is stored in the remaining bytes of the DTRx and its value is undefined.

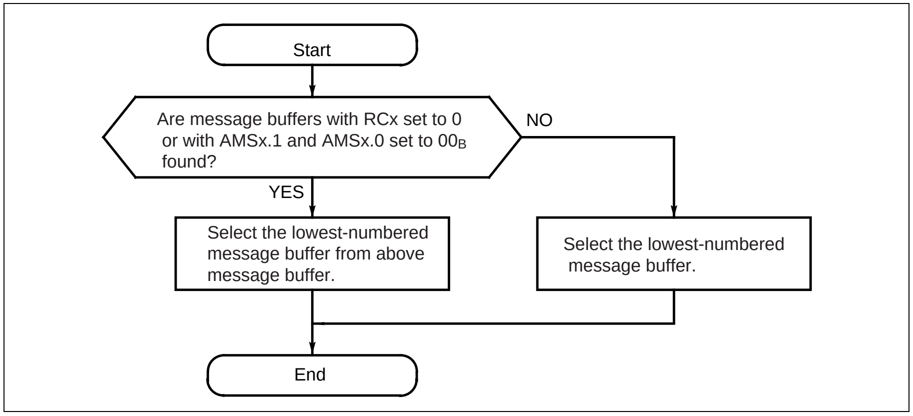
When receiving remote frames, received messages are stored only in the IDRx and DLCRx, and the DTRx remains unchanged.

If there is more than 1 message buffer including IDs passed through the acceptance filter, the message buffer x in which received messages are to be stored is determined according to the following rules.

- The order of priority of the message buffer x (x = 0 to 15) rises as its number lower; in other words, message buffer 0 is given the highest and the message buffer 15 is given the lowest priority.
- Basically, message buffers with the RCx bit of 0 in the receive completion register (RCR) are preferred in storing received messages.
- If the bits of the acceptance mask select register (AMSR) are set to All Bits Compare (for message buffers with the AMSx.1 and AMSx.0 bits set to 00_B), received messages are stored irrespective of the value of the RCx bit of the RCR.
- If there are message buffers with the RCx bit of the RCR set to 0, or with the bits of the AMSR set to All Bits Compare, received messages are stored in the lowest-number (highest-priority) message buffer x.
- If there are no message buffers above-mentioned, received messages are stored in a lower-number message buffer x.
- Message buffers should be arranged in ascending numeric order. The lowest message buffers should be with All Bits Compare, then AMR0 or AMR1 masks. And The highest message buffers should be with All Bits Mask.

Figure 21.6-1 shows a flowchart for determining the message buffer (x) where received messages are to be stored. It is recommended that message buffers be arranged in the following order: message buffers in which each AMSR bit is set to All Bits Compare, message buffers using AMR0 or AMR1, and message buffers in which each AMSR bit is set to All Bits Mask.

Figure 21.6-1 Flowchart Determining Message Buffer (x) where Received Messages Stored


Start
Are message buffers with RCx set to 0 or with AMSx.1 and AMSx.0 set to 00_B found?
NO
YES
Select the lowest-numbered message buffer from above message buffer.
Select the lowest-numbered message buffer.
End

■ Receive Overrun

When a message is stored in the message buffer with the corresponding RCx being already set to 1, it will results in receive overrun. In this case, the corresponding ROVRx bit in the receive overrun register ROVRR is set to 1.

■ Processing for Reception of Data Frame and Remote Frame

● Processing for reception of data frame

RRTRx of the remote request receiving register (RRTRR) becomes 0.

TREQx of the transmission request register (TREQR) becomes 0 (immediately before storing the received message). A transmission request for message buffer (x) having not executed transmission will be canceled.

Note:

A request for transmission of either a data frame or remote frame is canceled.

● Processing for reception of remote frame

RRTRx becomes 1.

If TRTRx of the transmitting RTR register (TRTRR) is 1, TREQx becomes 0. As a result, the request for transmitting remote frame to message buffer having not executed transmission will be canceled.

Notes:

- A request for data frame transmission is not canceled.
- For cancellation of a transmission request, see "21.5 Transmission of CAN Controller".

■ Completing Reception

RCx of the reception complete register (RCR) becomes 1 after storing the received message.

If a reception interrupt is enabled (RIEx of the reception interrupt enable register (RIER) is 1), an interrupt occurs.

Note:

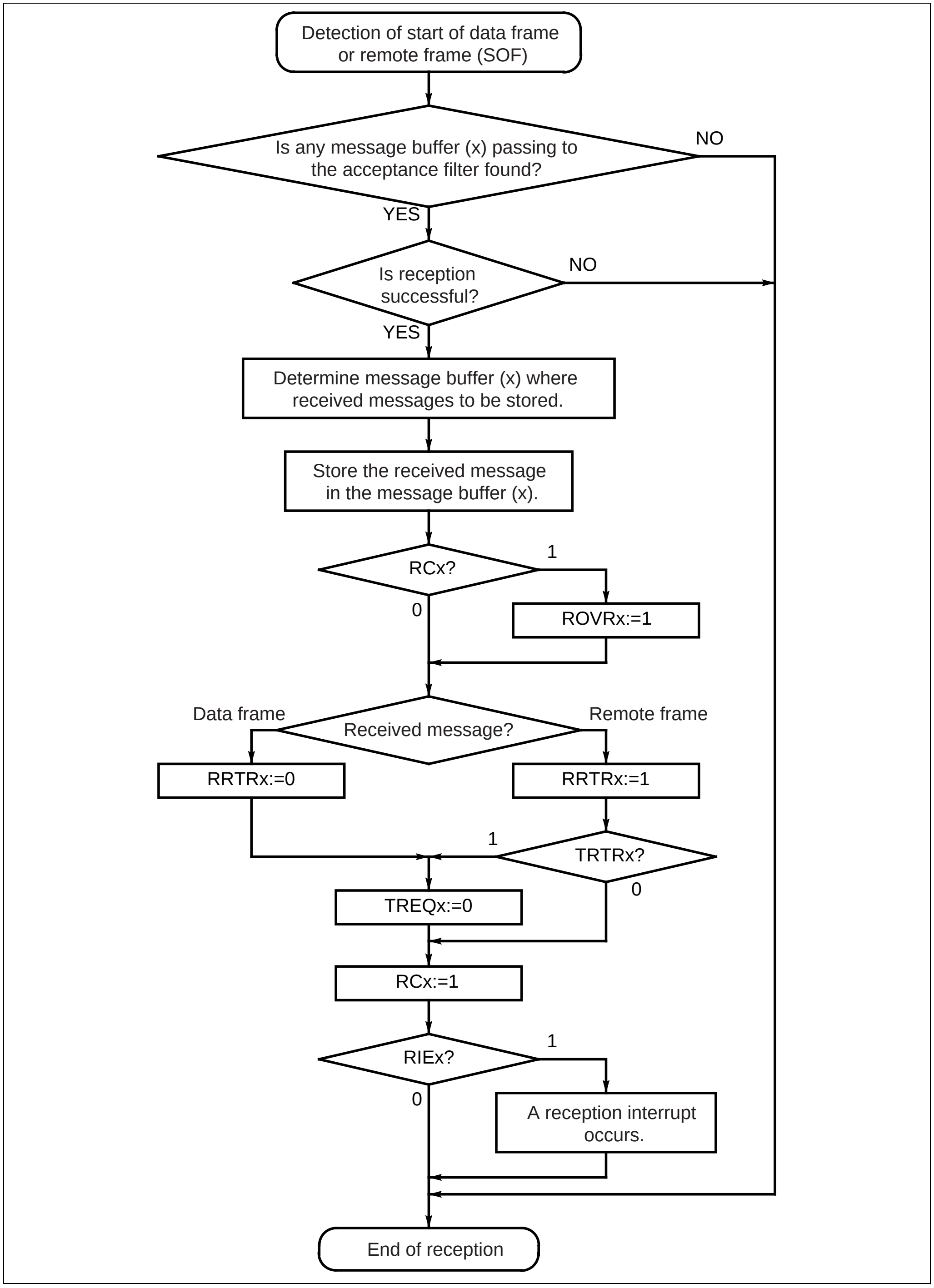
This CAN controller will not receive any messages transmitted by itself.

21.7 Reception Flowchart of CAN Controller

Figure 21.7-1 shows a reception flowchart of the CAN controller.

■ Reception Flowchart of the CAN Controller

Figure 21.7-1 Reception Flowchart of the CAN Controller


Detection of start of data frame or remote frame (SOF)
Is any message buffer (x) passing to the acceptance filter found?
NO
YES
Is reception successful?
NO
YES
Determine message buffer (x) where received messages to be stored.
Store the received message in the message buffer (x).
RCx?
1
0
ROVRx:=1
Data frame
Received message?
Remote frame
RRTRx:=0
RRTRx:=1
1
TRTRx?
0
TREQx:=0
RCx:=1
RIEx?
1
0
A reception interrupt occurs.
End of reception

21.8 How to Use CAN Controller

The following settings are required to use the CAN controller;

- **Bit timing**
- **Frame format**
- **ID**
- **Acceptance filter**
- **Low-power consumption mode**

■ Setting Bit Timing

The bit timing register (BTR) should be set during bus operation stop (when the bus operation stop bit (HALT) of the control status register (CSR) is 1).

After the setting completion, write 0 to HALT to cancel bus operation stop.

■ Setting Frame Format

Set the frame format used by the message buffer (x). When using the standard frame format, set IDEx of the IDE register (IDER) to 0. When using the extended frame format, set IDEx to 1.

This setting should be made when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

■ Setting ID

Set the message buffer (x) ID to ID28 to ID0 of ID register (IDRx). The message buffer (x) ID need not be set to ID11 to ID0 in the standard frame format. The message buffer (x) ID is used as a transmission message at transmission and is used as an acceptance code at reception.

This setting should be made when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

■ Setting Acceptance Filter

The acceptance filter of the message buffer (x) is set by an acceptance code and acceptance mask set. It should be set when the acceptance message buffer (x) is invalid (BVALx of the message buffer enable register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

Set the acceptance mask used in each message buffer (x) by the acceptance mask select register (AMSR). The acceptance mask registers (AMR0 and AMR1) should also be set if used (For the setting details, see "21.4.21 Acceptance Mask Select Register (AMSR)" and "21.4.22 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)").

The acceptance mask should be set so that a transmission request may not be canceled when unnecessary received messages are stored. For example, it should be set to a full-bit comparison if only one specific ID is used for the transmission.

■ Setting Low-power Consumption Mode

To set the F^2MC-16LX in a low-power consumption mode (Stop and Timebase timer), write 1 to the bus operation stop bit (HALT) of the control status register (CSR), and then check that the bus operation has stopped (HALT = 1).

21.9 Procedure for Transmission by Message Buffer (x)

After setting the bit timing, frame format, ID, and acceptance filter, set BVALx to 1 to activate the message buffer (x).

■ Procedure for Transmission by Message Buffer (x)

● Setting transmit data length code

Set the transmit data length code (byte count) to DLC3 to DLC0 of the DLC register (DLCRx).

For data frame transmission (when TRTRx of the transmission RTR register (TRTRR) is 0), set the data length of the transmitted message.

For remote frame transmission (when TRTRx = 1), set the data length (byte count) of the requested message.

Note:

Setting other than 0000_B to 1000_B (0 to 8 bytes) is prohibited.

● Setting transmit data (only for transmission of data frame)

For data frame transmission (when TRTRx of the transmission register (TRTRR) is 0), set data as the count of byte transmitted in the data register (DTRx).

Note:

Transmit data should be rewritten while the TREQx bit of the transmission request register (TREQR) set to 0. There is no need for setting the BVALx bit of the message buffer valid register (BVALR) to 0. Setting the BVALx bit to 0 may cause incoming remote frame to be lost.

● Setting transmission RTR register

For data frame transmission, set TRTRx of the transmission RTR register (TRTRR) to 0.

For remote frame transmission, set TRTRx to 1.

● Setting conditions for starting transmission (only for transmission of data frame)

Set RFWTx of the remote frame receiving wait register (RFWTR) to 0 to start transmission immediately after a request for data frame transmission is set (TREQx of the transmission request register (TREQR) is 1 and TRTRx of the transmission RTR register (TRTRR) is 0).

Set RFWTx to 1 to start transmission after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR) becomes 1) after a request for data frame transmission is set (TREQx = 1 and TRTRx = 0).

Note:

Remote frame transmission can not be made, if RFWTx is set to 1.

● Setting transmission complete interrupt

When generating a transmission complete interrupt, set TIEx of the transmission complete interrupt enable register (TIER) to 1.

When not generating a transmission complete interrupt, set TIEx to 0.

● Setting transmission request

For a transmission request, set TREQx of the transmission request register (TREQR) to 1.

● Canceling transmission request

When canceling a pending request for transmission to the message buffer (x), write 1 to TCANx of the transmission cancel register (TCANR).

Check TREQx. For TREQx = 0, transmission cancellation is terminated or transmission is completed. Check TCx of the transmission complete register (TCR). For TCx = 0, transmission cancellation is terminated. For TCx = 1, transmission is completed.

● Processing for completion of transmission

If transmission is successful, TCx of the transmission complete register (TCR) becomes 1.

If the transmission complete interrupt is enabled (TIEx of the transmission complete interrupt enable register (TIER) is 1), an interrupt occurs.

After checking the transmission completion, write 0 to TCx to set it to 0. This cancels the transmission complete interrupt.

In the following cases, the pending transmission request is canceled by receiving and storing a message.

- Cancel the request for data frame transmission by reception of data frame
- Cancel the request for remote frame transmission by reception of data frame
- Cancel the request for remote frame transmission by reception of remote frame

Request for data frame transmission is not canceled by receiving and storing a remote frame. ID and DLC, however, are changed by the ID and DLC of the received remote frame. Note that the ID and DLC of data frame to be transmitted become the value of received remote frame.

21.10 Procedure for Reception by Message Buffer (x)

After setting the bit timing, frame format, ID, and acceptance filter, make the settings described below.

■ Procedure for Reception by Message Buffer (x)

● Setting reception interrupt

To enable reception interrupt, set RIEx of the reception interrupt enable register (RIER) to 1.

To disable reception interrupt, set RIEx to 0.

● Starting reception

When starting reception after setting, set BVALx of the message buffer valid register (BVALR) to 1 to make the message buffer (x) valid.

● Processing for reception completion

If reception is successful after passing to the acceptance filter, the received message is stored in the message buffer (x) and RCx of the reception complete register (RCR) becomes 1. For data frame reception, RRTRx of the remote request receiving register (RRTRR) becomes 0. For remote frame reception, RRTRx becomes 1.

If a reception interrupt is enabled (RIEx of the reception interrupt enable register (RIER) is 1), an interrupt occurs.

After checking the reception completion (RCx = 1), process the received message.

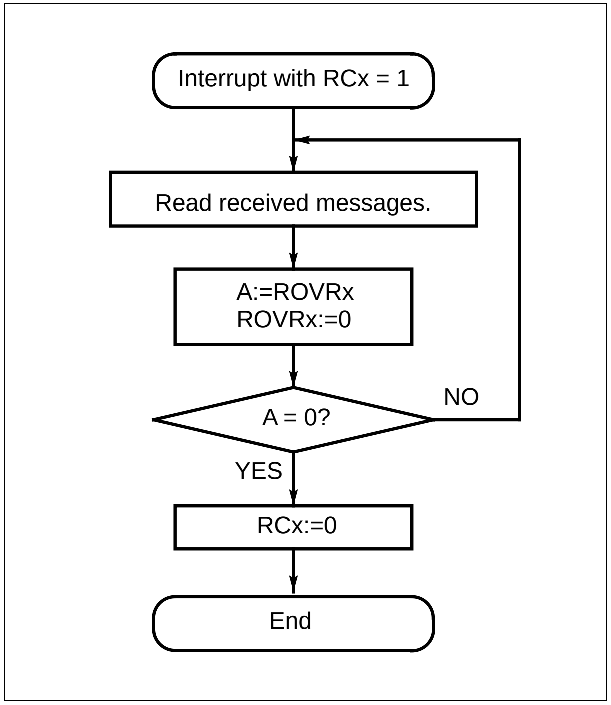
After completion of processing the received message, check ROVRx of the reception overrun register (ROVRR).

If ROVRx = 0, the processed received message is valid. Write 0 to RCRx to set it to 0 (the reception complete interrupt is also canceled) to terminate reception.

If ROVRx = 1, a reception overrun occurred and the next message may have overwritten the processed message. In this case, received messages should be processed again after setting the ROVRx bit to 0 by writing 0 to it.

Figure 21.10-1 shows an example of receive interrupt handling.

Figure 21.10-1 Example of Receive Interrupt Handling



21.11 Setting Configuration of Multi-level Message Buffer

If the receptions are performed frequently, or if several different ID’s of messages are received, in other words, if there is insufficient time for handling messages, more than 1 message buffer can be combined into a multi-level message buffer to provide allowance for processing time of the received message by CPU.

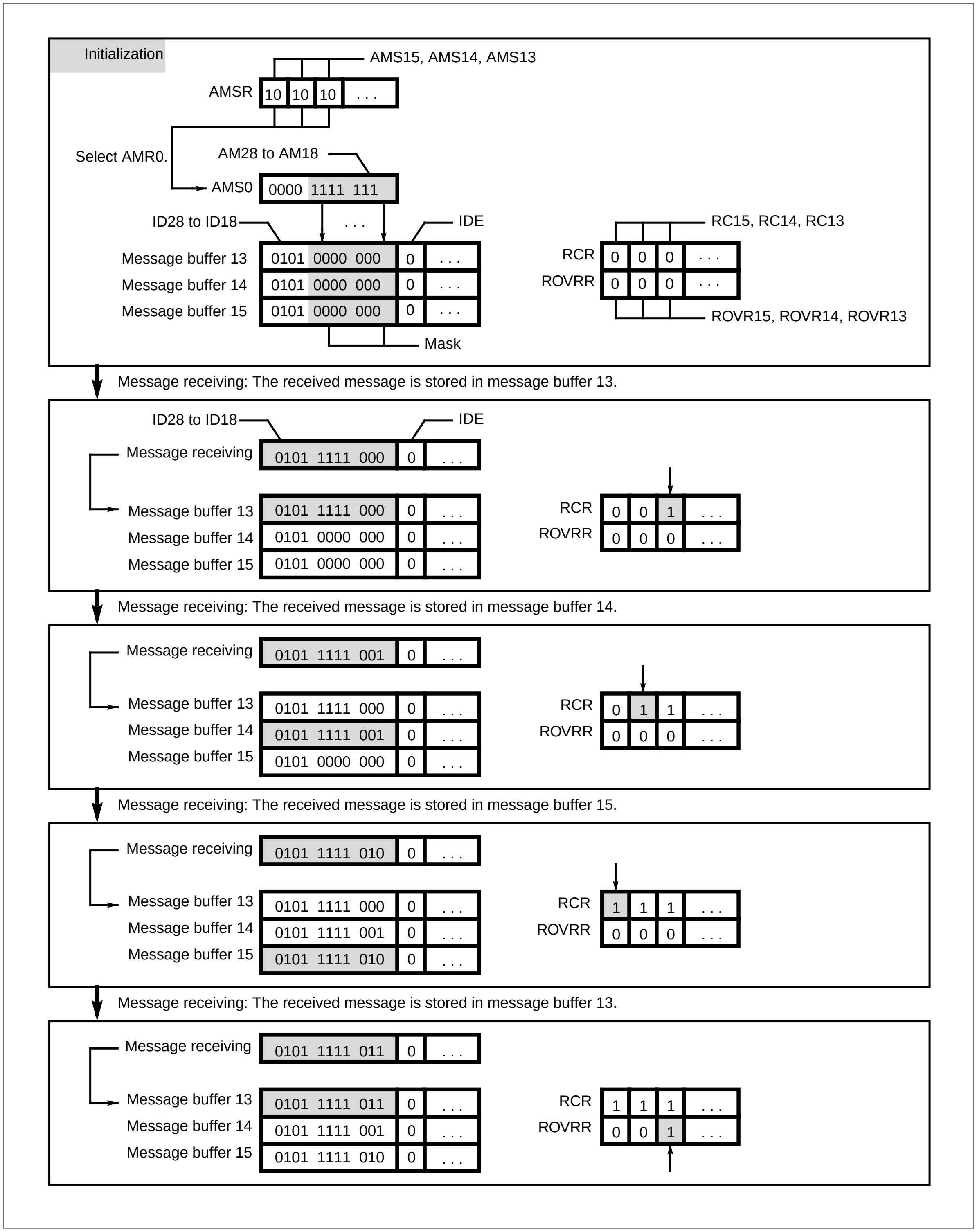
■ Setting Configuration of Multi-level Message Buffer

To provide a multi-level message buffer, the same acceptance filter must be set in the combined message buffers.

If the bits of the acceptance mask select register (AMSR) are set to All Bits Compare ((AMSx.1, AMSx.0) = (0, 0)), multi-level message configuration of message buffers is not allowed. This is because All Bits Compare causes received messages to be stored irrespective of the value of the RCx bit of the receive completion register (RCR), so received messages are always stored in lower-numbered (higher-priority) message buffers even if All Bits Compare and identical acceptance code (ID register (IDRx)) are specified for more than 1 message buffer. Therefore, All Bits Compare and identical acceptance code should not be specified for more than 1 message buffer.

Figure 21.11-1 shows operational examples of multi-level message buffers.

Figure 21.11-1 Examples of Operation of Multi-level Message Buffer



Note:

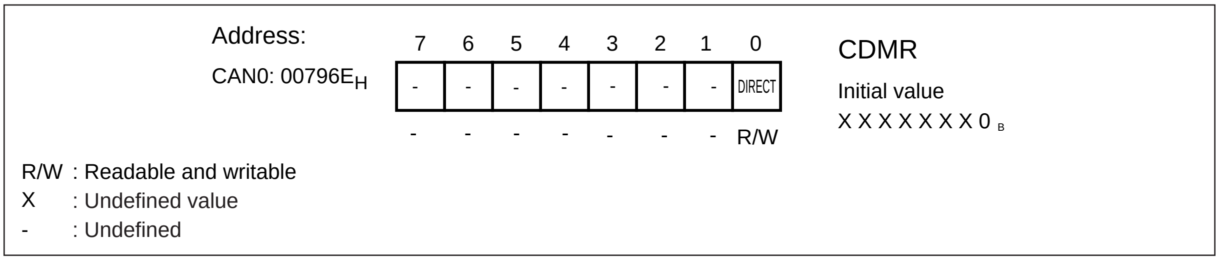
Four messages are received with the same acceptance filter set in message buffers 13, 14 and 15.

21.12 Setting the CAN Direct Mode Register

To operate CAN normally, this register must be set correctly.

■ CAN Direct Mode Register (CDMR) (Only MB90V340)

Figure 21.12-1 Configuration of the CAN Direct Mode Register (CDMR) (Only MB90V340)



R/W : Readable and writable
X : Undefined value
- : Undefined

Table 21.12-1 Function of CAN Direct Mode Register (CDMR)

Bit Name		Function
bit 7 to 1	Undefined bits	-
bit 0	DIRECT	If the clock modulation is set (initial state), the bit should be set "0". If the clock modulation is not set, the bit should be set "1".

Note:

MB90360 does not have the clock modulation function.
So, at using CAN controller, the DIRECT bit of the register must be set "1".

21.13 Precautions when Using CAN Controller

Use of the CAN Controller requires the following cautions.

■ Caution for Disabling Message Buffers by BVAL Bits

The use of BVAL bits may affect malfunction of CAN Controller when messages buffers are set disabled while CAN Controller is participating in CAN communication. This section shows the work around of this malfunction.

● Condition

When following 2 conditions occur at the same time, the CAN Controller will not perform to transmit messages normally.

- CAN Controller is participating in the CAN communication. (i.e. The read value of the CSR: HALT bit is 0 and CAN Controller is ready to transmit messages)
- Message buffers are read when BVAL bits disable the message buffers.

● Work around

Operation for suppressing transmission request

Do not use BVAL bit for suppressing transmission request, use TCAN bit instead of it.

Operation for composing transmission message

For composing a transmission message, it is necessary to disable the message buffer by BVAL bit to change contents of ID and IDE registers. In this case, BVAL bit should reset (BVAL=0) after checking if TREQ bit is 0 or after completion of the previous message transmission (TC=1).

In case a buffer needs to be disabled, ensure that no transmission request is pending (if it was requested before). Therefore, do not reset BVALx-Bit before testing, if a transmission is ongoing;

a) Cancel the transmission request (TCANx=1;), if necessary

b) and wait for the transmission completion (while (TREQx==1);) by polling or interrupt.

Only after that the transmission buffer can be disabled (BVALx=0;).

Note:

For case a), if transmission of that buffer has already started, canceling the request is ignored and disabling the buffer is delayed until the end of the transmission.

■ Setting of CAN Direct Mode

MB90360 does not provide the clock modulation function. For this reason, ensure that the DIRECT bit of the CAN direct mode register (CDMR) is set to 1 when CAN is used.

Note that the CAN controller will not normally operate without correct setting of the DIRECT bit.

CHAPTER 22

ADDRESS MATCH DETECTION FUNCTION

This chapter explains the address match detection function and its operation.

22.1 Overview of Address Match Detection Function

If the address of the instruction to be processed next to the instruction currently processed by the program matches the address set in the detect address setting registers, the address match detection function forcibly replaces the next instruction to be processed by the program with the INT9 instruction to branch to the interrupt processing program. Since the address match detection function can use the INT9 interrupt for instruction processing, the program can be corrected by patch processing.

■ Overview of Address Match Detection Function

- The address of the instruction to be processed next to the instruction currently processed by the program is always held in the address latch through the internal data bus. The address match detection function always compares the value of the address held in the address latch with that of the address set in the detect address setting registers. When these compared values match, the next instruction to be processed by the CPU is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.
- There are six detect address setting registers (PADR0 to PADR5), each of which has an interrupt enable bit. The generation of an interrupt due to a match between the address held in the address latch and the address set in the detect address setting registers can be enabled or disabled for each register.

22.2 Block Diagram of Address Match Detection Function

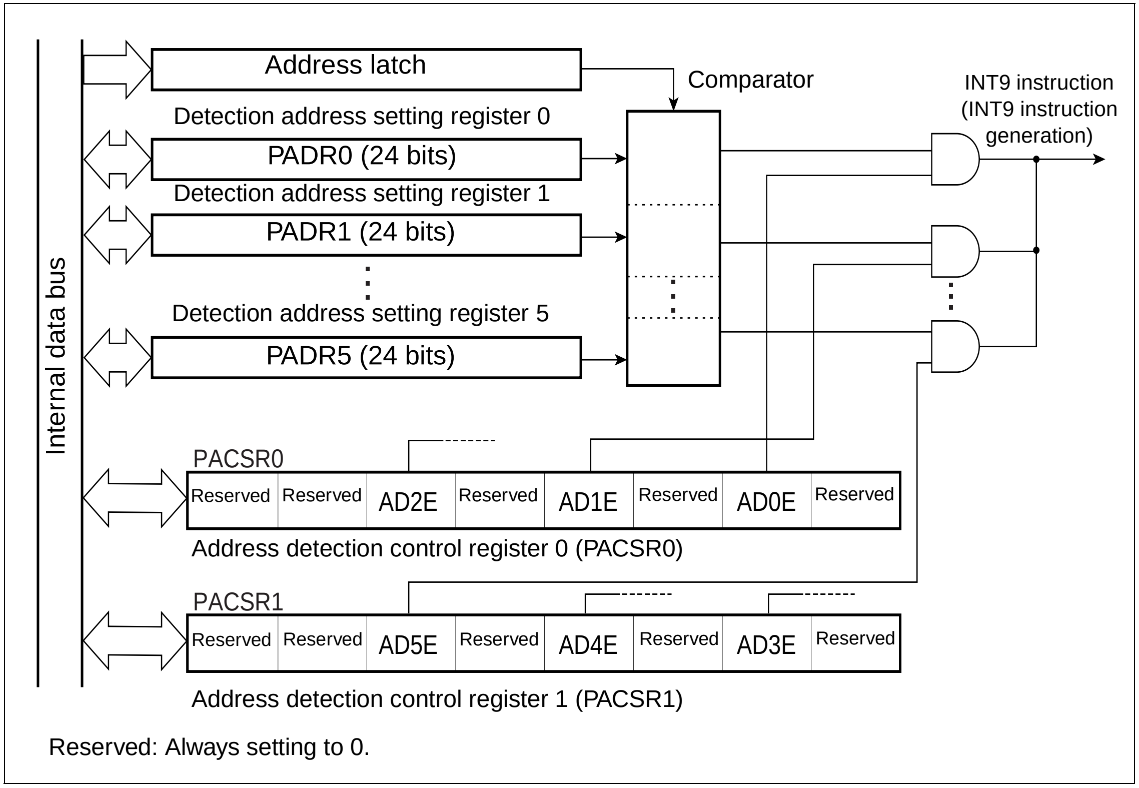
The address match detection module consists of the following blocks:

- **Address latch**
- **Address detection control register (PACSR0/PACSR1)**
- **Detect address setting registers (PADR0 to PADR5)**

■ Block Diagram of Address Match Detection Function

Figure 22.2-1 shows the block diagram of the address match detection function.

Figure 22.2-1 Block Diagram of the Address Match Detection Function



● Address latch

The address latch stores the value of the address output to the internal data bus.

● Address detection control register (PACSR0/PACSR1)

The address detection control register enables or disables output of an interrupt at an address match.

● Detect address setting registers (PADR0 to PADR5)

The detect address setting registers set the address that is compared with the value of the address latch.

22.3 Configuration of Address Match Detection Function

This section lists and details the registers used by the address match detection function.

■ List of Registers and Reset Values of Address Match Detection Function

Figure 22.3-1 List of Registers and Reset Values of Address Match Detection Function

	bit	7	6	5	4	3	2	1	0
Address detection control register 0(PACSR0)		0	0	0	0	0	0	0	0
	bit	15	14	13	12	11	10	9	8
Address detection control register 1(PACSR1)		0	0	0	0	0	0	0	0
	bit	7	6	5	4	3	2	1	0
Detection address setting register 0(PADR0): Low		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 0(PADR0): Middle		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 0(PADR0): High		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 1(PADR1): Low		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 1(PADR1): Middle		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 1(PADR1): High		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 2(PADR2): Low		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 2(PADR2): Middle		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 2(PADR2): High		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 3(PADR3): Low		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 3(PADR3): Middle		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 3(PADR3): High		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 4(PADR4): Low		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 4(PADR4): Middle		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 4(PADR4): High		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 5(PADR5): Low		×	×	×	×	×	×	×	×
	bit	15	14	13	12	11	10	9	8
Detection address setting register 5(PADR5): Middle		×	×	×	×	×	×	×	×
	bit	7	6	5	4	3	2	1	0
Detection address setting register 5(PADR5): High		×	×	×	×	×	×	×	×

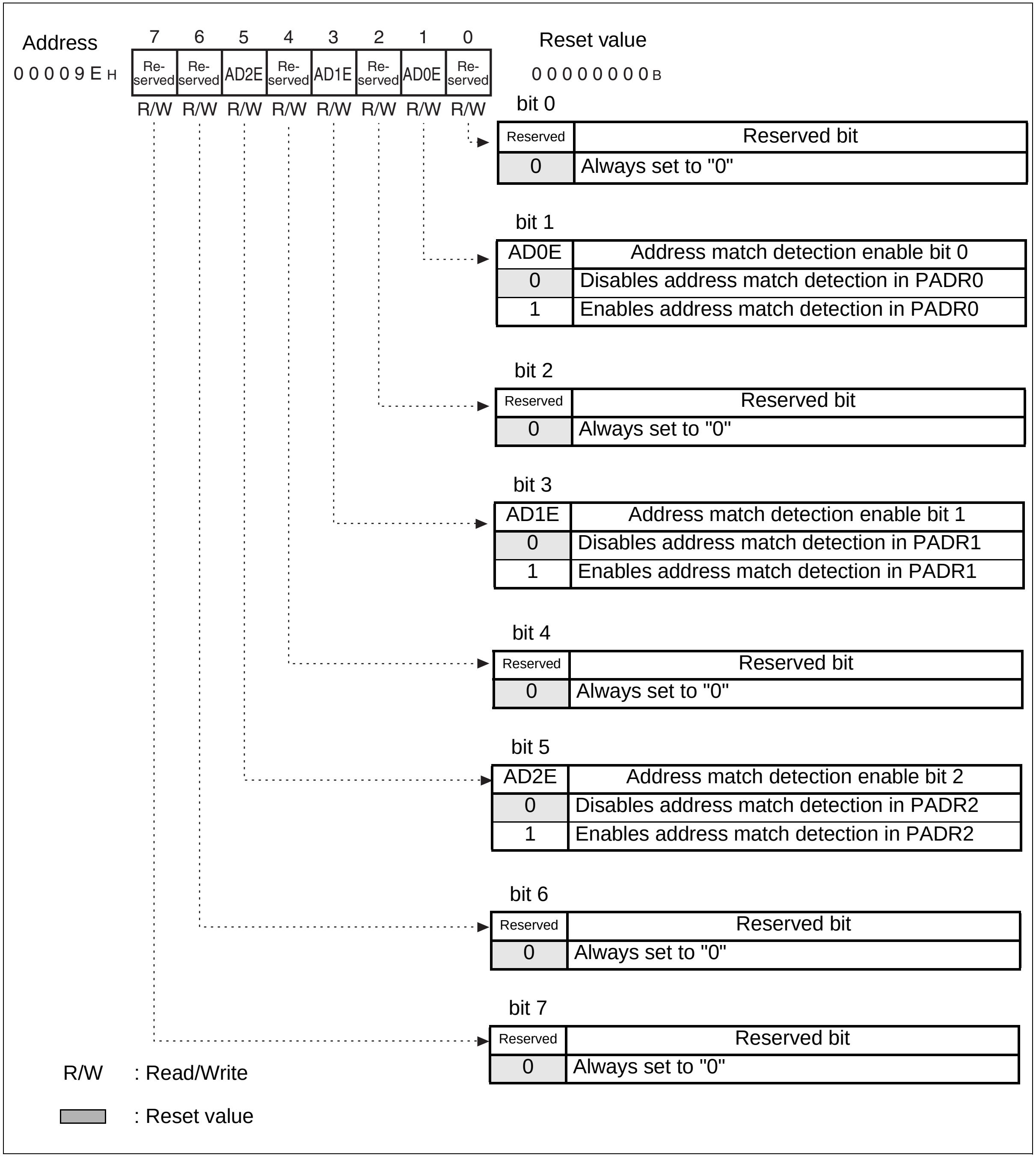
×: Undefined

22.3.1 Address Detection Control Register (PACSR0/PACSR1)

The address detection control register enables or disables output of an interrupt at an address match. When an address match is detected when output of an interrupt at an address match is enabled, the INT9 interrupt is generated.

■ Address Detection Control Register 0 (PACSR0)

Figure 22.3-2 Address Detection Control Register 0 (PACSR0)



Address	7	6	5	4	3	2	1	0	Reset value
0 0 0 0 9 E H	Re-served	Re-served	AD2E	Re-served	AD1E	Re-served	AD0E	Re-served	0 0 0 0 0 0 0 0 B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

bit 0

Reserved	Reserved bit
0	Always set to "0"

bit 1

AD0E	Address match detection enable bit 0
0	Disables address match detection in PADR0
1	Enables address match detection in PADR0

bit 2

Reserved	Reserved bit
0	Always set to "0"

bit 3

AD1E	Address match detection enable bit 1
0	Disables address match detection in PADR1
1	Enables address match detection in PADR1

bit 4

Reserved	Reserved bit
0	Always set to "0"

bit 5

AD2E	Address match detection enable bit 2
0	Disables address match detection in PADR2
1	Enables address match detection in PADR2

bit 6

Reserved	Reserved bit
0	Always set to "0"

bit 7

Reserved	Reserved bit
0	Always set to "0"

R/W : Read/Write

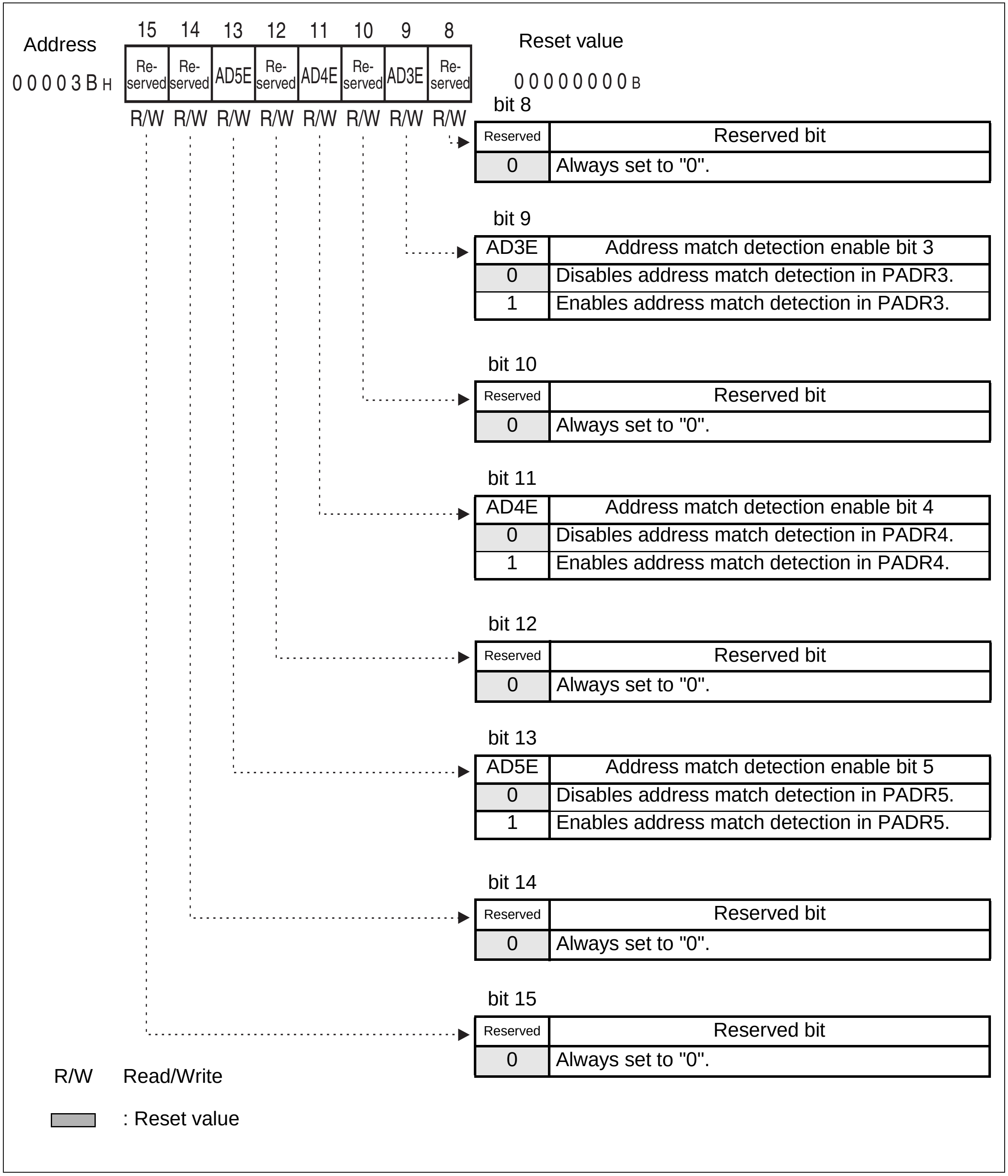
▨ : Reset value

Table 22.3-1 Functions of Address Detection Control Register (PACSR0)

Bit Name		Function
bit7, bit6	Reserved: reserved bits	Always set to 0.
bit5	AD2E: Address match detection enable bit 2	The address match detection operation with the detect address setting register 2 (PADR2) is enabled or disabled. **When set to 0:** Disables the address match detection operation. **When set to 1:** Enables the address match detection operation. • When the value of detect address setting registers 2 (PADR2) matches with the value of address latch at enabling the address match detection operation (AD2E = 1), the INT9 instruction is immediately executed.
bit4	Reserved: reserved bit	Always set to 0.
bit3	AD1E: Address match detection enable bit 1	The address match detection operation with the detect address setting register 1 (PADR1) is enabled or disabled. **When set to 0:** Disables the address match detection operation. **When set to 1:** Enables the address match detection operation. • When the value of detect address setting registers 1 (PADR1) matches with the value of address latch at enabling the address match detection operation (AD1E = 1), the INT9 instruction is immediately executed.
bit2	Reserved: reserved bit	Always set to 0.
bit1	AD0E: Address match detection enable bit 0	The address match detection operation with the detect address setting register 0 (PADR0) is enabled or disabled. **When set to 0:** Disables the address match detection operation. **When set to 1:** Enables the address match detection operation. • When the value of detect address setting registers 0 (PADR0) matches with the value of address latch at enabling the address match detection operation (AD0E = 1), the INT9 instruction is immediately executed.
bit0	Reserved: reserved bit	Always set to 0.

■ Address Detection Control Register 1 (PACSR1)

Figure 22.3-3 Address Detection Control Register 1 (PACSR1)


Address
00003BH
15 14 13 12 11 10 9 8
Re-served Re-served AD5E Re-served AD4E Re-served AD3E Re-served
R/W R/W R/W R/W R/W R/W R/W R/W
Reset value
00000000B

bit 8

Reserved	Reserved bit
0	Always set to "0".

bit 9

AD3E	Address match detection enable bit 3
0	Disables address match detection in PADR3.
1	Enables address match detection in PADR3.

bit 10

Reserved	Reserved bit
0	Always set to "0".

bit 11

AD4E	Address match detection enable bit 4
0	Disables address match detection in PADR4.
1	Enables address match detection in PADR4.

bit 12

Reserved	Reserved bit
0	Always set to "0".

bit 13

AD5E	Address match detection enable bit 5
0	Disables address match detection in PADR5.
1	Enables address match detection in PADR5.

bit 14

Reserved	Reserved bit
0	Always set to "0".

bit 15

Reserved	Reserved bit
0	Always set to "0".

R/W Read/Write

▨ : Reset value

Table 22.3-2 Functions of Address Detection Control Register (PACSR1)

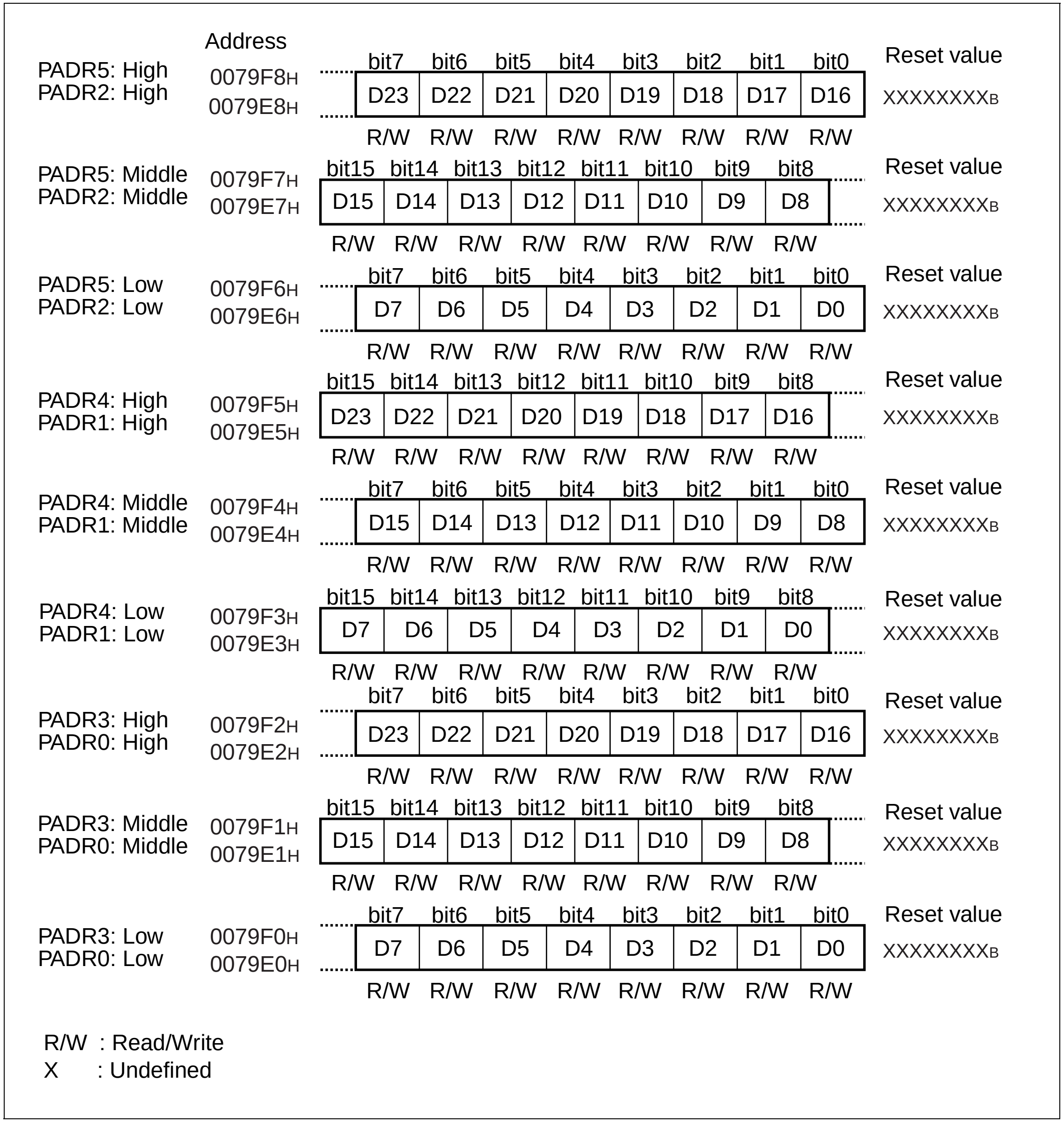
Bit Name		Function
bit15, bit14	Reserved: reserved bit	Always set to 0.
bit13	AD5E: Address match detection enable bit 5	The address match detection operation with the detect address setting register 5 (PADR5) is enabled or disabled. **When set to 0:** Disables the address match detection operation. **When set to 1:** Enables the address match detection operation. • When the value of detect address setting registers 5 (PADR5) matches with the value of address latch at enabling the address match detection operation (AD5E = 1), the INT9 instruction is immediately executed.
bit12	Reserved: reserved bit	Always set to 0.
bit11	AD4E: Address match detection enable bit 4	The address match detection operation with the detect address setting register 4 (PADR4) is enabled or disabled. **When set to 0:** Disables the address match detection operation. **When set to 1:** Enables the address match detection operation. • When the value of detect address setting registers 4 (PADR4) matches with the value of address latch at enabling the address match detection operation (AD4E = 1), the INT9 instruction is immediately executed.
bit10	Reserved: reserved bit	Always set to 0.
bit9	AD3E: Address match detection enable bit 3	The address match detection operation with the detect address setting register 3 (PADR3) is enabled or disabled. **When set to 0:** Disables the address match detection operation. **When set to 1:** Enables the address match detection operation. • When the value of detect address setting registers 3 (PADR3) matches with the value of address latch at enabling the address match detection operation (AD3E = 1), the INT9 instruction is immediately executed.
bit8	Reserved: reserved bit	Always set to 0.

22.3.2 Detect Address Setting Registers (PADR0 to PADR5)

The value of an address to be detected is set in the detect address setting registers. When the address of the instruction processed by the program matches the address set in the detect address setting registers, the next instruction is forcibly replaced by the INT9 instruction, and the interrupt processing program is executed.

■ Detect Address Setting Registers (PADR0 to PADR5)

Figure 22.3-4 Detect Address Setting Registers (PADR0 to PADR5)



	Address									Reset value
		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
PADR5: High PADR2: High	0079F8H 0079E8H	D23	D22	D21	D20	D19	D18	D17	D16	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
PADR5: Middle PADR2: Middle	0079F7H 0079E7H	D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
PADR5: Low PADR2: Low	0079F6H 0079E6H	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
PADR4: High PADR1: High	0079F5H 0079E5H	D23	D22	D21	D20	D19	D18	D17	D16	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
PADR4: Middle PADR1: Middle	0079F4H 0079E4H	D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
PADR4: Low PADR1: Low	0079F3H 0079E3H	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
PADR3: High PADR0: High	0079F2H 0079E2H	D23	D22	D21	D20	D19	D18	D17	D16	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
PADR3: Middle PADR0: Middle	0079F1H 0079E1H	D15	D14	D13	D12	D11	D10	D9	D8	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
		bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
PADR3: Low PADR0: Low	0079F0H 0079E0H	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXXB
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W : Read/Write
X : Undefined

■ Functions of Detect Address Setting Registers

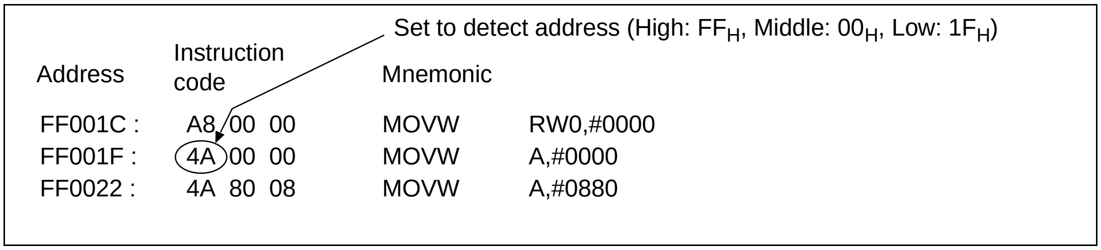
- There are six detect address setting registers (PADR0 to PADR5) that consist of a high byte (bank), middle byte, and low byte, totaling 24 bits.

Table 22.3-3 Address Setting of Detect Address Setting Registers

Register Name	Interrupt Output Enable	Address Setting	
Detect address setting register 0 (PADR0)	PACSR0: AD0E	High	Set the upper 8 bits of detect address 0 (bank).
		Middle	Set the middle 8 bits of detect address 0.
		Low	Set the lower 8 bits of detect address 0.
Detect address setting register 1 (PADR1)	PACSR0: AD1E	High	Set the upper 8 bits of detect address 1 (bank).
		Middle	Set the middle 8 bits of detect address 1.
		Low	Set the lower 8 bits of detect address 1.
Detect address setting register 2 (PADR2)	PACSR0: AD2E	High	Set the upper 8 bits of detect address 2 (bank).
		Middle	Set the middle 8 bits of detect address 2.
		Low	Set the lower 8 bits of detect address 2.
Detect address setting register 3 (PADR3)	PACSR1: AD3E	High	Set the upper 8 bits of detect address 3 (bank).
		Middle	Set the middle 8 bits of detect address 3.
		Low	Set the lower 8 bits of detect address 3.
Detect address setting register 4 (PADR4)	PACSR1: AD4E	High	Set the upper 8 bits of detect address 4 (bank).
		Middle	Set the middle 8 bits of detect address 4.
		Low	Set the lower 8 bits of detect address 4.
Detect address setting register 5 (PADR5)	PACSR1: AD5E	High	Set the upper 8 bits of detect address 5 (bank).
		Middle	Set the middle 8 bits of detect address 5.
		Low	Set the lower 8 bits of detect address 5.

- In the detect address setting registers (PADR0 to PADR5), starting address (first byte) of instruction to be replaced by INT9 instruction should be set.

Figure 22.3-5 Setting of Starting Address of Instruction Code to be Replaced by INT9 Instruction


Set to detect address (High: FF$_H$, Middle: 00$_H$, Low: 1F$_H$)

```
Address    Instruction      Mnemonic
           code

FF001C :    A8  00  00      MOVW      RW0,#0000
FF001F :    4A  00  00      MOVW      A,#0000
FF0022 :    4A  80  08      MOVW      A,#0880
```

Notes:

- When an address of other than the first byte is set to the detect address setting registers (PADR0 to PADR5), the instruction code is not replaced by INT9 instruction and a program of an interrupt processing is not be performed. When the address is set to the second byte or subsequent, the address set by the instruction code is replaced by "01" (INT9 instruction code) and, which may cause malfunction.
- The detect address setting registers (PADR0 to PADR5) should be set after disabling the address match detection (PACSR: ADnE=0) of corresponding address match control registers. If the detect address setting registers are changed without disabling the address match detection, the address match detection function will work immediately after an address match occurs during writing address, which may cause malfunction.
- The address match detection function can be used only for addresses of the internal ROM area. If addresses of the external memory area are set, the address match detection function will not work and the INT9 instruction will not be executed.

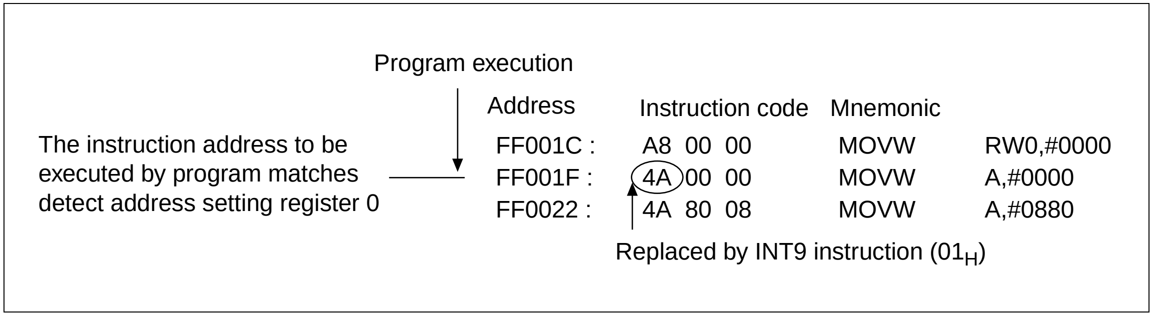
22.4 Explanation of Operation of Address Match Detection Function

If the addresses of the instructions executed in the program match those set in the detection address setting registers (PADR0 to PADR5), the address match detection function will replace the first instruction code executed by the CPU with the INT9 instruction (01_H) to branch to the interrupt processing program.

■ Operation of Address Match Detection Function

Figure 22.4-1 shows the operation of the address match detection function when the detect addresses are set and an address match is detected.

Figure 22.4-1 Operation of Address Match Detection Function



```
Address      Instruction code   Mnemonic
FF001C :     A8  00  00          MOVW       RW0,#0000
FF001F :     4A  00  00          MOVW       A,#0000
FF0022 :     4A  80  08          MOVW       A,#0880
```

Program execution
The instruction address to be executed by program matches detect address setting register 0
Replaced by INT9 instruction (01_H)

■ Setting Detect Address

1) Disable the detection address setting register 0 (PADR0) where the detect address is set for address match detection (PACSR0: AD0E=0).

2) Set the detect address in the detection address setting register 0 (PADR0). Set "FF_H" at the higher bits, "00_H" at the middle bits, and "$1F_H$" at the lower bits of the detection address setting register 0 (PADR0).

3) Enable the detect address setting register 0 (PADR0) where the detect address is set for address match detection (PACSR0: AD0E=1).

■ Program Execution

1) If the address of the instruction to be executed in the program matches the set detect address, the first instruction code at the matched address is replaced by the INT9 instruction code ("01_H").

2) INT9 instruction is executed. INT9 interrupt is generated and then interrupt processing program is executed.

22.4.1 Example of using Address Match Detection Function

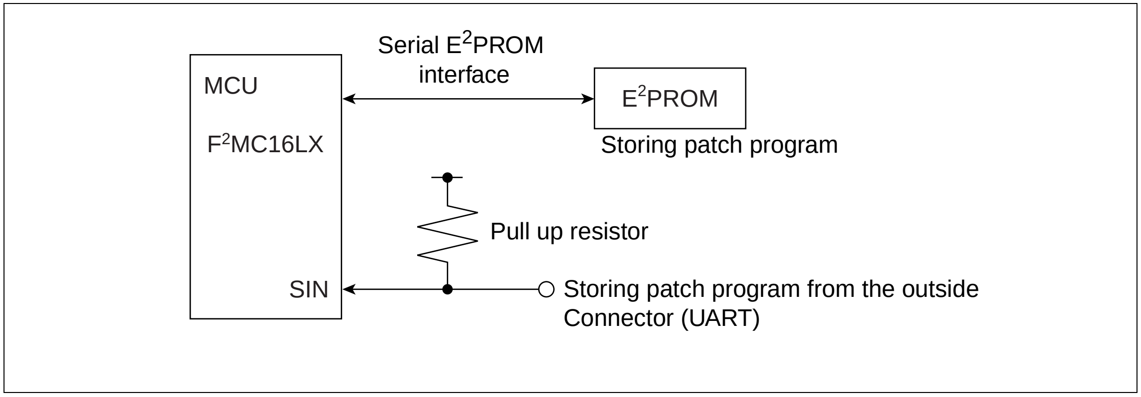
This section gives an example of patch processing for program correction using the address match detection function.

■ System Configuration and E^{2}PROM Memory Map

● System configuration

Figure 22.4-2 gives an example of the system configuration using the address match detection function.

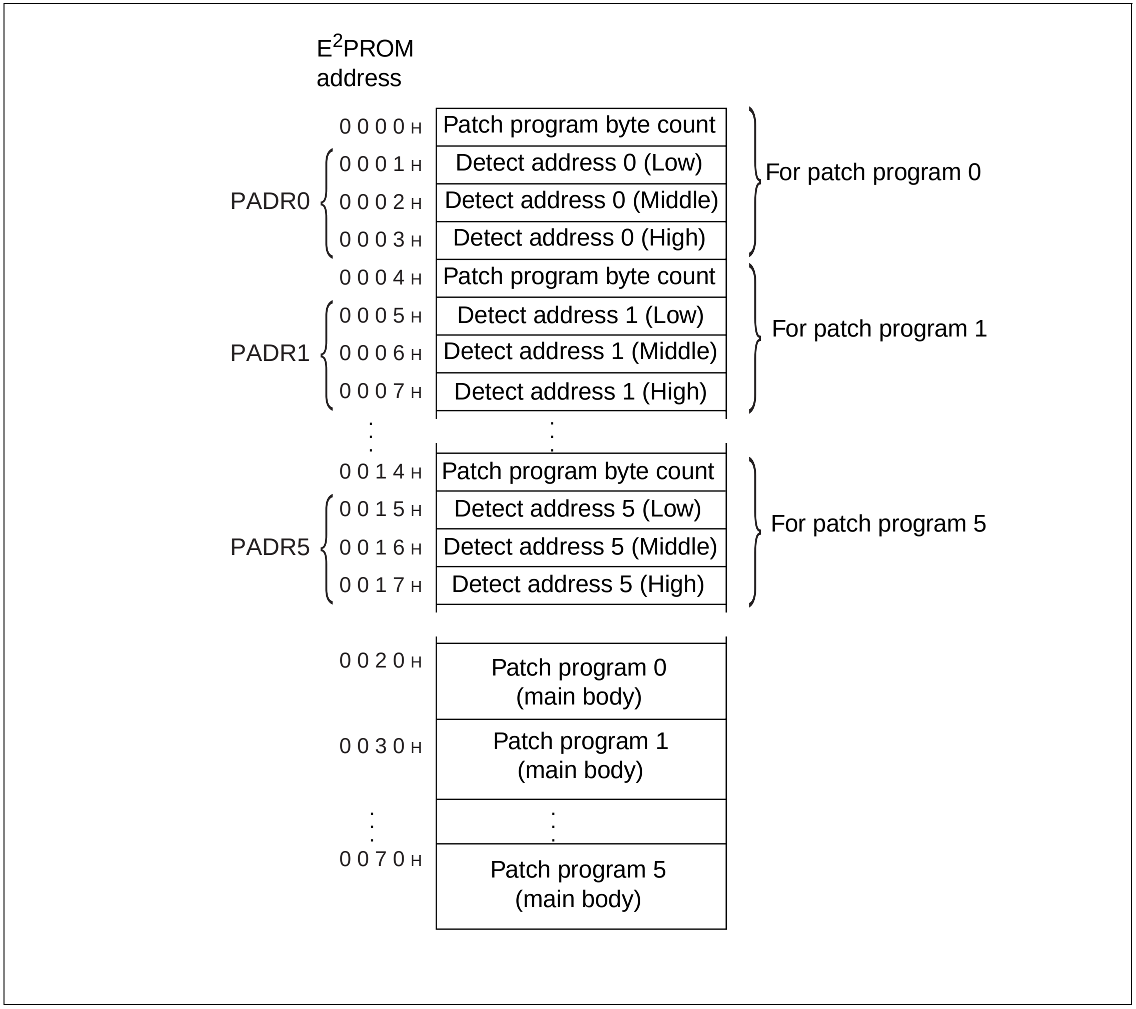
Figure 22.4-2 Example of System Configuration Using Address Match Detection Function



■ E^{2}PROM Memory Map

Figure 22.4-3 shows the allocation of the patch program and data at storing the patch program in E^{2}PROM.

Figure 22.4-3 Allocation of E^{2}PROM Patch Program and Data



● Patch program byte count

The total byte count of the patch program (main body) is stored. If the byte count is "00_H", it indicates that no patch program is provided.

● Detect address (24 bits)

The address where the instruction code is replaced by the INT9 instruction code due to program error is stored. This address is set in the detection address setting registers (PADR0 to PADR5).

● Patch program (main body)

The program executed by the INT9 interrupt processing when the program address matches the detect address is stored. Patch program 0 is allocated from any predetermined address. Patch program 1 is allocated from the address indicating <starting address of patch program 0 + total byte count of patch program 0>.
It is similar for the correction program 2 to 5.

■ Setting and Operating State

● Initialization

E^2PROM data are all cleared to "00_H".

● Occurrence of program error

- By using the connector (UART), information about the patch program is transmitted to the MCU (F^2MC16LX) from the outside according to the allocation of the E^2PROM patch program and data.
- The MCU (F^2MC16LX) stores the information received from outside in the E^2PROM.

● Reset sequence

- After reset, the MCU (F^2MC16LX) reads the byte count of the E^2PROM patch program to check the presence or absence of the correction program.
- If the byte count of the patch program is not "00_H", the higher, middle and lower bits at detect addresses 0 to 5 are read and set in the detection address setting registers 0 to 5 (PADR0 to PADR5). The patch program (main body) is read according to the byte count of the patch program and written to RAM in the MCU (F^2MC16LX).
- The patch program (main body) is allocated to the address where the patch program is executed in the INT9 interrupt processing by the address match detection function.
- Address match detection is enabled (PACSR: AD0E=1, AD1E=1 ... AD5E=1).

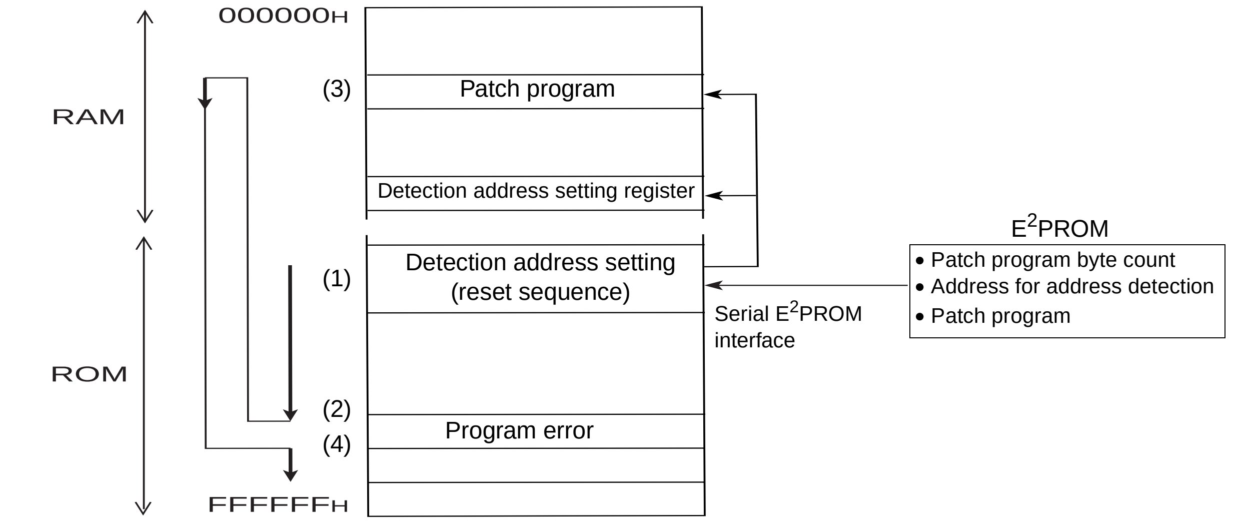
● INT9 Interrupt processing

- Interrupt processing is performed by the INT9 instruction. The MB90360 series has no interrupt request flag by address match detection. Therefore, if the stack information in the program counter is discarded, the detect address cannot be checked. When checking the detect address, check the value of program counter stacked in the interrupt processing routine.
- The patch program is executed, branching to the normal program.

■ Operation of Address Match Detection Function at Storing Patch Program in E^{2}PROM

Figure 22.4-4 shows the operation of the address match detection function at storing the patch program in E^{2}PROM.

Figure 22.4-4 Operation of Address Match Detection Function at Storing Patch Program in E^{2}PROM

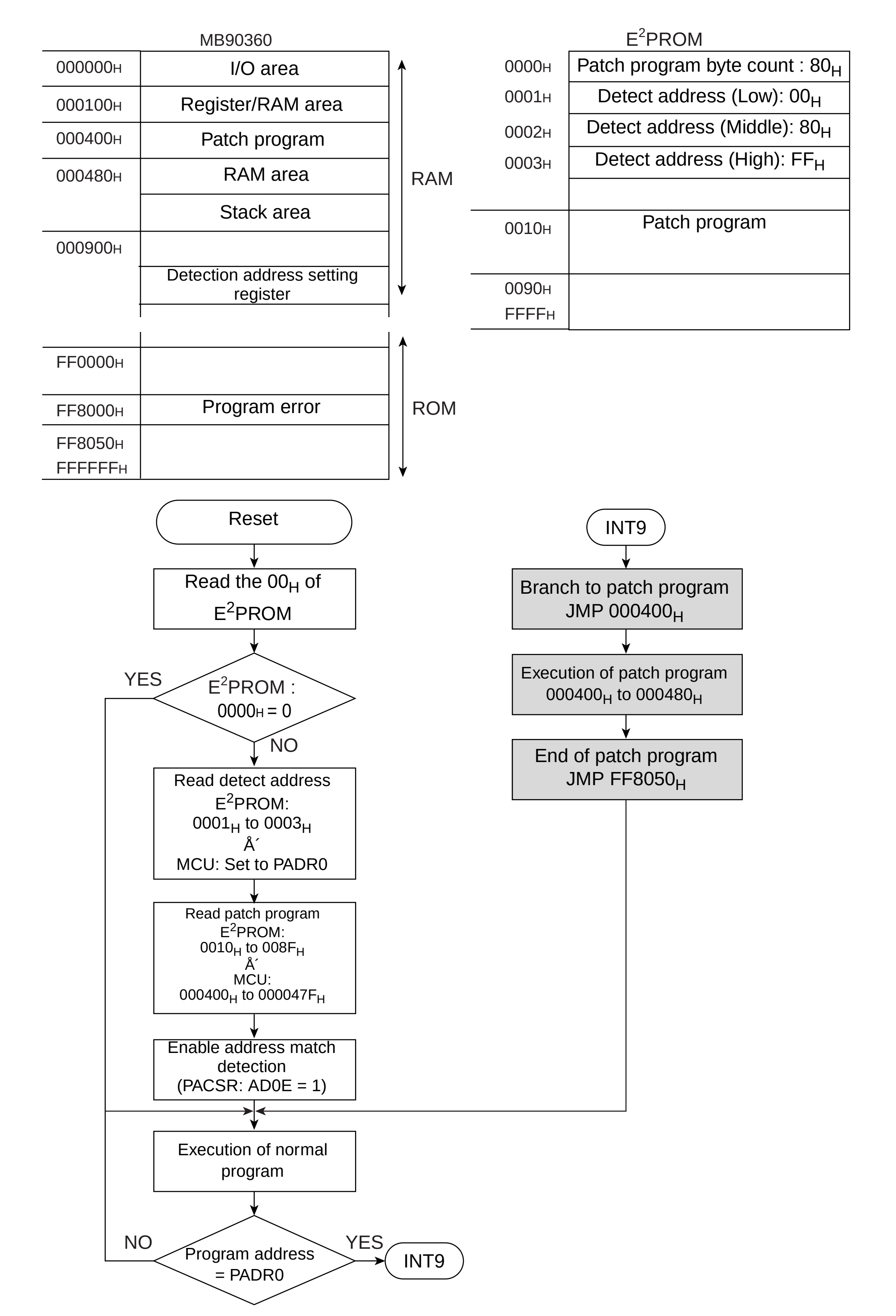


(1) Execution of detection address setting of reset sequence and normal program
(2) Branch to patch program which expanded in RAM with INT9 interrupt processing by address match detection
(3) Patch program execution by branching of INT9 processing
(4) Execution of normal program which branches from patch program

■ Flow of Patch Processing for Patch Program

Figure 22.4-5 shows the flow of patch processing for patch program using the address match detection function.

Figure 22.4-5 Flow of Patch Processing for Patch Program


MB90360
000000H
I/O area
000100H
Register/RAM area
000400H
Patch program
000480H
RAM area
Stack area
000900H
Detection address setting register
RAM
FF0000H
FF8000H
Program error
FF8050H
FFFFFFH
ROM
E2PROM
0000H
Patch program byte count : 80H
0001H
Detect address (Low): 00H
0002H
Detect address (Middle): 80H
0003H
Detect address (High): FFH
0010H
Patch program
0090H
FFFFH
Reset
Read the 00H of E2PROM
E2PROM : 0000H = 0
YES
NO
Read detect address E2PROM: 0001H to 0003H Å´ MCU: Set to PADR0
Read patch program E2PROM: 0010H to 008FH Å´ MCU: 000400H to 00047FH
Enable address match detection (PACSR: AD0E = 1)
Execution of normal program
Program address = PADR0
NO
YES
INT9
INT9
Branch to patch program JMP 000400H
Execution of patch program 000400H to 000480H
End of patch program JMP FF8050H

22.5 Program Example of Address Match Detection Function

This section gives a program example for the address match detection function.

■ Program Example for Address Match Detection Function

- Processing specifications

If the address of the instruction to be executed by the program matches the address set in the detection address setting register (PADR0), the INT9 instruction is executed.

- Coding example

```
PACSR0 EQU   00009EH           ;Address detection control register 0
PADRL  EQU   0079E0H           ;Detection address setting register 0
                               (Low)
PADRM  EQU   0079E1H           ;Detection address setting register 0
                               (Middle)
PADRH  EQU   0079E2H           ;Detection address setting register 0
                               (High)
;
;---------Main program-------------------------------------
CODE   CSEG
START:
                               ;Stack pointer (SP), etc.,
                               ;already initialized
       MOV   PADRL,#00H        ;Set address detection register 0
                               (Low)
       MOV   PADRM,#00H        ;Set address detection register 0
                               (Middle)
       MOV   PADRH,#00H        ;Set address detection register 0
                               (High)
;
       MOV   I:PACSR0,#00000010B ;Enable address match
       .
       processing by user
       .
LOOP:
       .
       processing by user
       .
       BRA   LOOP
;---------Interrupt program-------------------------------------
WARI:
       .
       processing by user
       .
       RETI                    ;Return from interrupt processing
CODE   ENDS
;---------Vector setting-----------------------------------------
VECT   CSEG  ABS=0FFH
       ORG   00FFD8H
       DSL   WARI
       ORG   00FFDCH           ;Set reset vector
       DSL   START
       DB    00H               ;Set to single-chip mode
VECT   ENDS
       END   START
```

CHAPTER 23
ROM MIRRORING MODULE

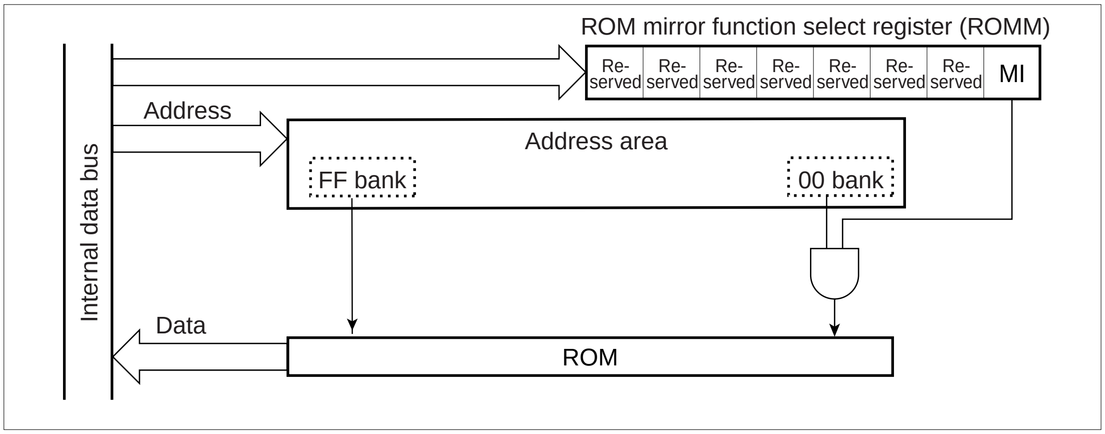
This chapter describes the functions and operations of the ROM mirroring function select module.

23.1 Overview of ROM Mirroring Function Select Module

The ROM mirroring function select module provides a setting so that ROM data in the FF bank can be read by access to the 00 bank.

■ Block Diagram of ROM Mirroring Function Select Module

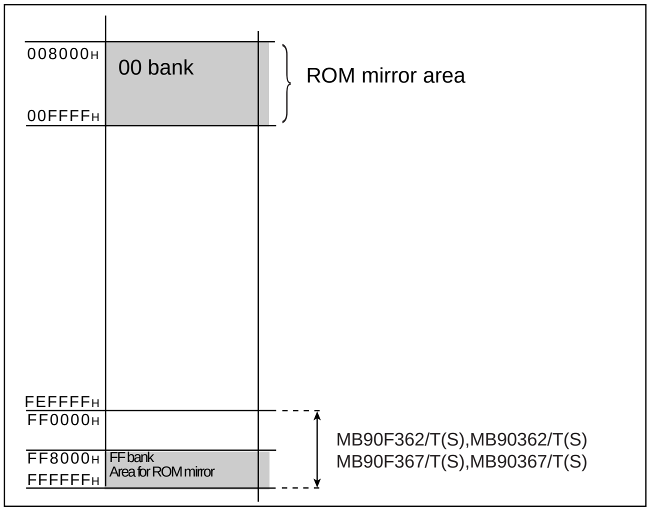
Figure 23.1-1 Block Diagram of ROM Mirroring Function Select Module



■ Access to FF Bank by ROM Mirroring Function

Figure 23.1-2 shows the location in memory when ROM mirroring function allows access to the 00 bank to read ROM data in the FF bank.

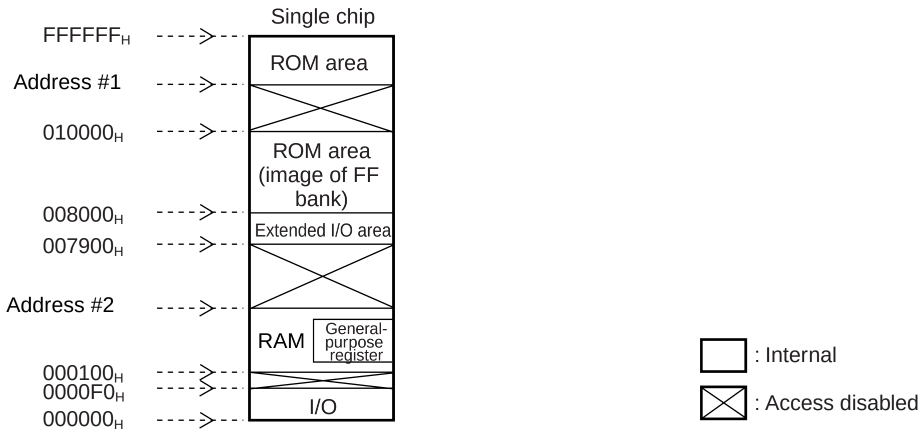
Figure 23.1-2 Access to FF Bank by ROM Mirroring Function



■ Memory Space when ROM Mirroring Function Enabled/Disabled

Figure 23.1-3 shows the availability of access to memory space when the ROM mirroring function is enabled or disabled.

Figure 23.1-3 Memory Space when ROM Mirroring Function Enabled/Disabled



Product type	Address #1	Address #2
MB90F362/T(S), MB90362/T(S), MB90F367/T(S), MB90367/T(S)	FF0000$_H$	000D00$_H$
MB90V340A-101/102/103/104	F80000$_H$	007900$_H$

■ List of Registers and Reset Values of ROM Mirroring Function Select Module

Figure 23.1-4 List of Registers and Reset Values of ROM Mirroring Function Select Module

bit	15	14	13	12	11	10	9	8
ROM mirror function select register (ROMM)	×	×	×	×	×	×	×	1

× : Undefined

23.2 ROM Mirroring Function Select Register (ROMM)

The ROM mirroring function select register (ROMM) enables or disables the ROM mirroring function. When the ROM mirroring function is enabled, ROM data in the FF bank can be read by access to the 00 bank.

■ ROM Mirroring Function Select Register (ROMM)

Figure 23.2-1 ROM Mirroring Function Select Register (ROMM)

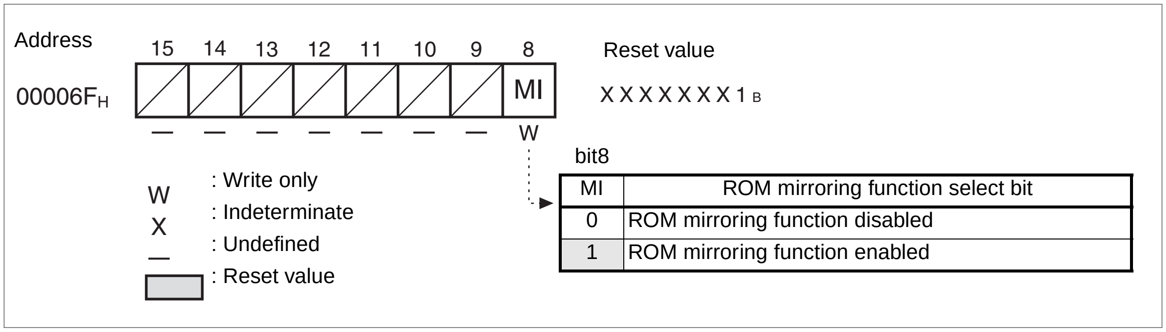


Table 23.2-1 Functions of ROM Mirroring Function Select Register (ROMM)

Bit Name		Function
bit8	MI: ROM mirroring function select bit	This bit enables or disables the ROM mirroring function. **When set to 0:** Disables ROM mirroring function **When set to 1:** Enables ROM mirroring function • When the ROM mirroring function is enabled (MI = 1), data at ROM addresses "FF8000$_H$" to "FFFFFF$_H$" can be read by accessing addresses "008000$_H$" to "00FFFF$_H$".
bit9 to bit15	Undefined bits	**Read:** Value is undefined. **Write:** No effect

Note:

While the ROM area at addresses "008000$_H$" to "00FFFF$_H$" is being used, access to the ROM mirroring function select register (ROMM) is prohibited.

CHAPTER 24

512K-BIT FLASH MEMORY

This chapter explains the functions and operation of the 512K-bit flash memory. The following three methods are available for writing data to and erasing data from the flash memory:

- **Parallel programmer**
- **Serial programmer**
- **Executing programs to write/erase data**

This chapter explains “Executing programs to write/erase data”.

24.1 Overview of 512K-bit Flash Memory

The 512K-bit flash memory is mapped to the FF$_H$ bank in the CPU memory map. The functions of the flash memory interface circuit enable read-access and program-access from the CPU in the same way as mask ROM. Instructions from the CPU can be used via the flash memory interface circuit to write data to and erase data from the flash memory. Internal CPU control therefore enables rewriting of the flash memory while it is mounted. As a result, improvements in programs and data can be performed efficiently.

■ 512K-bit Flash Memory Features

- Use of automatic program algorithm (Embedded Algorithm^{TM*}: Equivalent to MBM29LV200)
- Detection of completion of writing/erasing using data polling or toggle bit functions
- Detection of completion of writing/erasing using CPU interrupts
- Minimum of 10,000 write/erase operations
- Flash reading cycle time: Minimum of 2 machine cycles

*: Embedded Algorithm^{TM} is a trademark of Advanced Micro Devices, Inc.

Note:

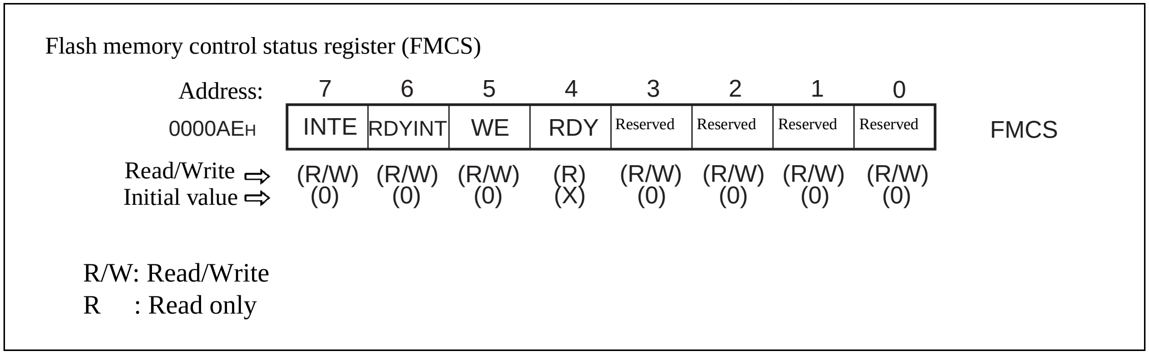
The manufacturer code and device code do not have the reading function. These codes cannot be accessed by the command.

■ Writing to/erasing Flash Memory

The flash memory cannot be written to and erased at the same time. That is, when data is written to or erased data from the flash memory, the program in the flash memory must first be copied to RAM. The entire process is then executed in RAM so that data is simply written to the flash memory. This eliminates the need for the program to access the flash memory from the flash memory itself.

■ Flash Memory Control Status Register (FMCS)

Figure 24.1-1 Flash Memory Control Status Register (FMCS)



Flash memory control status register (FMCS)

Address:	7	6	5	4	3	2	1	0	
0000AE$_H$	INTE	RDYINT	WE	RDY	Reserved	Reserved	Reserved	Reserved	FMCS
Read/Write ⇨	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)	

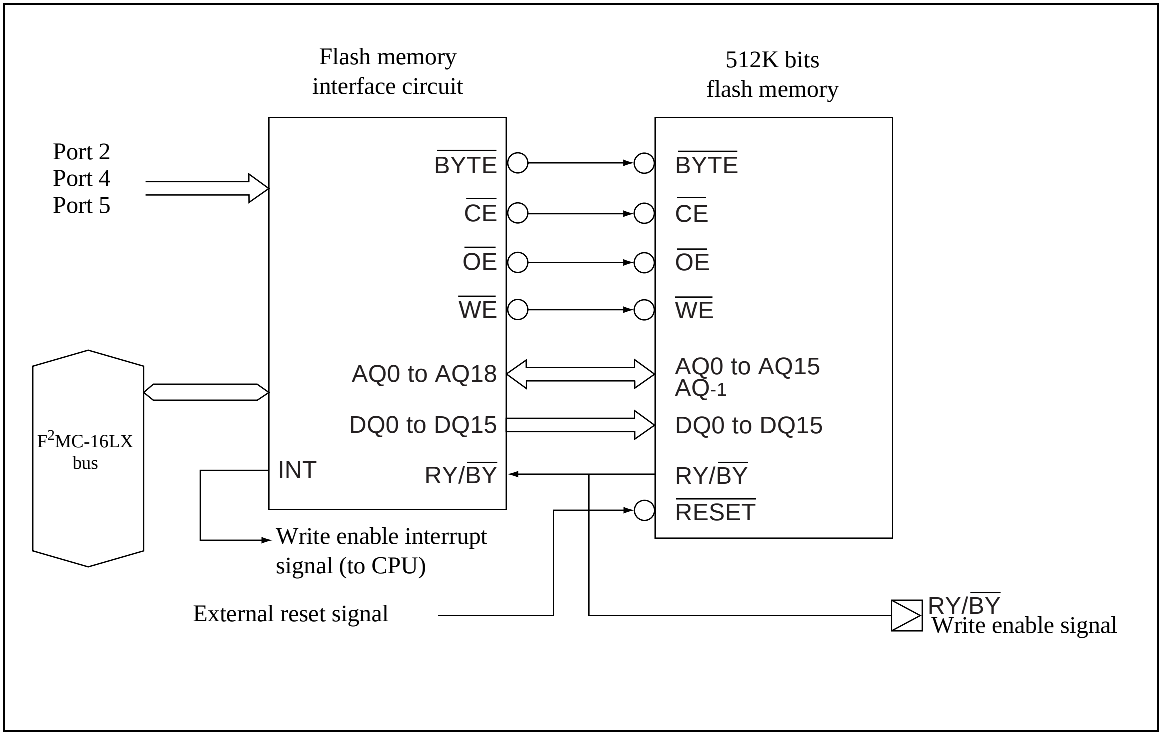
R/W: Read/Write
R : Read only

24.2 Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory

Figure 24.2-1 shows a block diagram of the entire flash memory with the flash memory interface circuit included. Figure 24.2-2 shows the sector configuration of the flash memory.

■ Block Diagram of the Entire Flash Memory

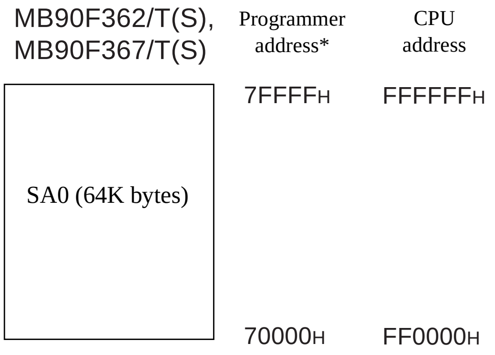
Figure 24.2-1 Block Diagram of the Entire Flash Memory



■ Sector Configuration of the 512K-bit Flash Memory

Figure 24.2-2 shows the sector configuration of the 512K-bit flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

Figure 24.2-2 Sector Configuration of the 512K-bit Flash Memory



*: The programmer address is equivalent to the CPU address when data is written to the flash memory using a parallel programmer. When a general programmer is used for writing/erasing, this address is used for writing/erasing.

24.3 Write/Erase Modes

The flash memory can be accessed in 2 different ways: Flash memory mode and alternative mode. Flash memory mode enables data to be directly written to or erased from the external pins. Alternative mode enables data to be written to or erased from the CPU via the internal bus. Use the mode external pins to select the mode.

■ Flash Memory Mode

The CPU stops when the mode pins are set to "111_B" while the reset signal is asserted. The flash memory interface circuit is connected directly to ports 2, 4 and 5, enabling direct control from the external pins. This mode makes the MCU seem like a standard flash memory to the external pins, and write/erase can be performed using a flash memory programmer.

In flash memory mode, all operations supported by the flash memory automatic algorithm can be used.

■ Alternative Mode

The flash memory is located in the FF bank in the CPU memory space, and like ordinary mask ROM, can be read-accessed and program-accessed from the CPU via the flash memory interface circuit.

Since writing/erasing the flash memory is performed by instructions from the CPU via the flash memory interface circuit, this mode allows rewriting even when the MCU is soldered on the target board.

■ Flash Memory Control Signals

Table 24.3-1 lists the flash memory control signals in flash memory mode.

The flash memory control signals and the external pin of the MBM29LV200 have one-to-one relationship.

In flash memory mode, the external data bus signal width is limited to 8 bits, enabling only 1 byte access. The DQ15 to DQ8 pins are not supported. The $\overline{\text{BYTE}}$ pin should always be set to 0.

Table 24.3-1 Flash Memory Control Signals (Developing: it is possible to change)

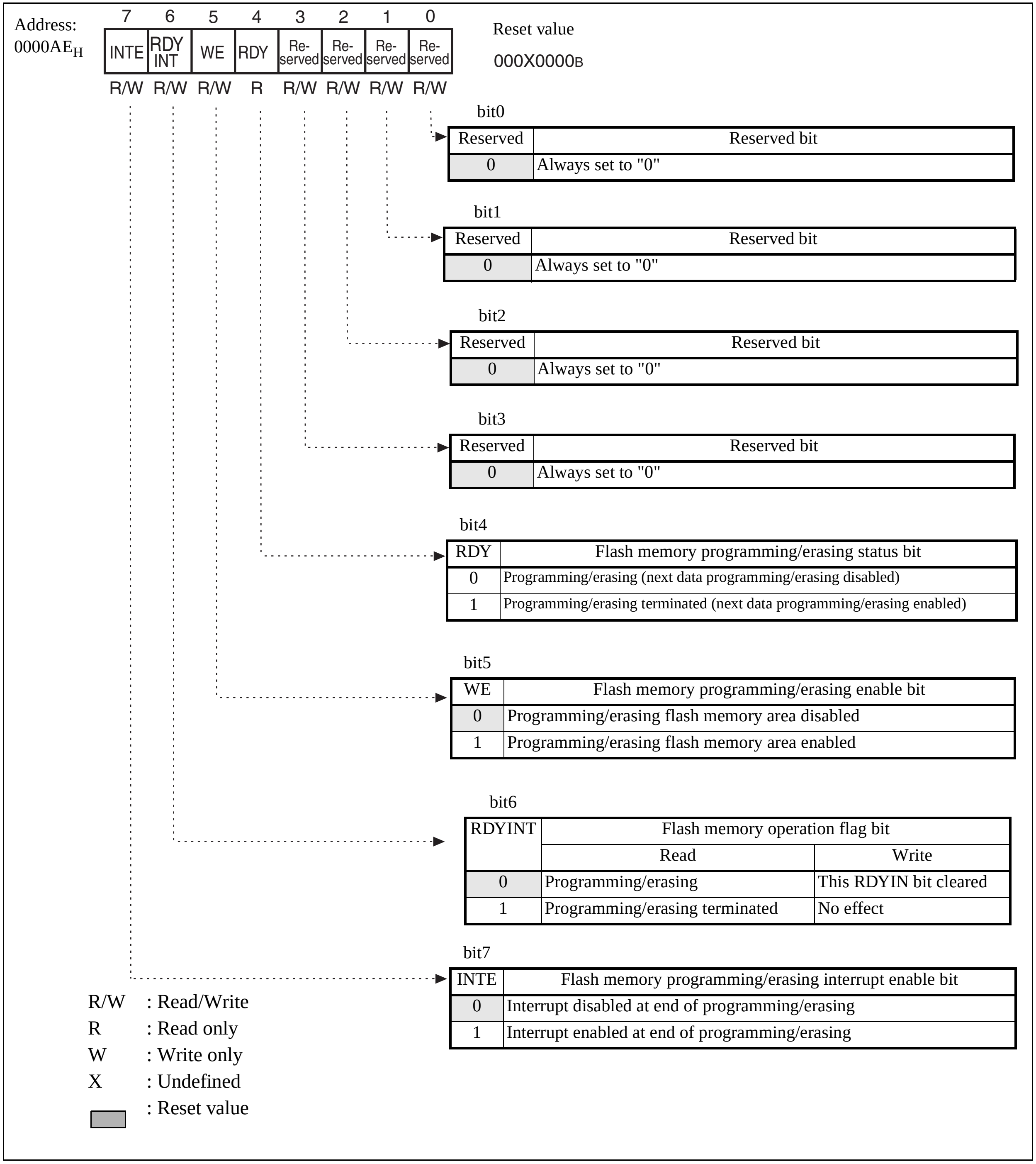
MB90F362/T(S), MB90F367/T(S)			MBM29LV200
Pin number	Normal function	Flash memory mode	
LQFP			
42	P83	AQ16	A15
38	P87	$\overline{\text{CE}}$	$\overline{\text{CE}}$
39	P86	$\overline{\text{OE}}$	$\overline{\text{OE}}$
40	P43	$\overline{\text{WE}}$	$\overline{\text{WE}}$
41 (45)	P42(P44)	AQ17 (AQ18)	A16
37	P85	$\overline{\text{BYTE}}$	$\overline{\text{BYTE}}$
11	P80	RY/$\overline{\text{BY}}$	RY/$\overline{\text{BY}}$
12 to 19	P50 to P57	AQ8 to AQ15	A7 to A14
21	MD1	MD1	$\overline{\text{RESET}}$(V_{ID})
20	MD2	MD2	OE(V_{ID})
3 to 10	P60 to P67	DQ0 to DQ7	DQ0 to DQ7
23	$\overline{\text{RST}}$	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$
29 to 36	P27 to P20	AQ0 to AQ7	A-1, A0 to A6

24.4 Flash Memory Control Status Register (FMCS)

This section shows the function of the flash memory control status register (FMCS).

■ Flash Memory Control Status Register (FMCS)

Figure 24.4-1 Flash Memory Control Status Register (FMCS)



Address: 0000AE$_H$

7	6	5	4	3	2	1	0
INTE	RDY INT	WE	RDY	Re-served	Re-served	Re-served	Re-served
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Reset value

000X0000$_B$

bit0

Reserved	Reserved bit
0	Always set to "0"

bit1

Reserved	Reserved bit
0	Always set to "0"

bit2

Reserved	Reserved bit
0	Always set to "0"

bit3

Reserved	Reserved bit
0	Always set to "0"

bit4

RDY	Flash memory programming/erasing status bit
0	Programming/erasing (next data programming/erasing disabled)
1	Programming/erasing terminated (next data programming/erasing enabled)

bit5

WE	Flash memory programming/erasing enable bit
0	Programming/erasing flash memory area disabled
1	Programming/erasing flash memory area enabled

bit6

RDYINT	Flash memory operation flag bit	
	Read	Write
0	Programming/erasing	This RDYIN bit cleared
1	Programming/erasing terminated	No effect

bit7

INTE	Flash memory programming/erasing interrupt enable bit
0	Interrupt disabled at end of programming/erasing
1	Interrupt enabled at end of programming/erasing

R/W : Read/Write

R : Read only

W : Write only

X : Undefined

▨ : Reset value

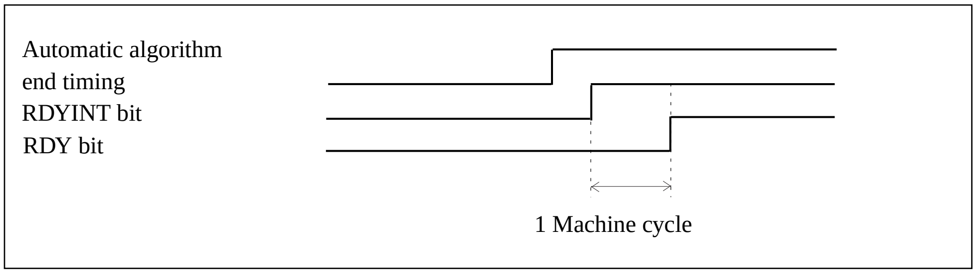
Table 24.4-1 Functions of Control Status Register (FMCS)

Bit Name		Function
bit7	INTE: Flash memory programming/erasing interrupt enable bit	This bit enables or disables an interrupt as programming/erasing flash memory is terminated. **When set to 1:** If the flash memory operation flag bit is set to 1 (FMCS: RDYINT=1), an interrupt is requested.
bit6	RDYINT: Flash memory operation flag bit	This bit shows the operating state of flash memory. If programming/erasing flash memory is terminated, the RDYINT bit is set to 1 in timing of termination of the flash memory automatic algorithm. • If the RDYINT bit is set to 1 when an interrupt as programming/erasing flash memory is terminated is enabled (FMCS:INTE = 1), an interrupt is requested. • If the RDYINT bit is 0, programming/erasing flash memory is disabled. **When set to 0:** Cleared. **When set to 1:** No effect If the read-modify-write (RMW) instructions are used, 1 is always read.
bit5	WE: Flash memory programming/erasing enable bit	This bit enables or disables the programming/erasing of flash memory. The WE bit should be set before starting the command to program/erase flash memory. **When set to 0:** No program/erase signal is generated even if the command to program/erase the FF bank is inputted. **When set to 1:** Programming/erasing flash memory is enabled after inputting program/erase command to the FF bank. When not performing programming/erasing, the WE bit should be set to 0 so as not to accidentally program or erase flash memory.
bit4	RDY: Flash memory programming/erasing status bit	This bit shows the programming/erasing status of flash memory. If the RDY bit is 0, programming/erasing flash memory is disabled. The read/reset command can be accepted even if the RDY bit is 0. The RDY bit is set to 1 when programming/erasing is completed.
bit3 to bit0	Reserved: Reserved bits	Always set to 0.

Note:

- The RDYINT and RDY bits cannot be changed at the same time. Create a program so that decisions are made using one or the other of these bits. (See Figure 24.4-2 .)
- This register can be accessed only in byte-access mode.

Figure 24.4-2 Transitions of the RDYINT and RDY Bits



24.5 Starting the Flash Memory Automatic Algorithm

Three types of commands are available for starting the flash memory automatic algorithm: Read/Reset, Write, and Chip Erase.

■ Command Sequence Table

Table 24.5-1 lists the commands used for flash memory write/erase. All of the data written to the command register is in bytes, but use word access to write. The data of the high-order bytes at this time is ignored.

Table 24.5-1 Command Sequence Table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/ reset*	1	FFXXXX	XXF0	-	-	-	-	-	-	-	-	-	-
Read/ reset*	4	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XXF0	RA	RD	-	-	-	-
Write program	4	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip erase	6	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XX80	FFAAAA	XXAA	FF5554	XX55	FFAAAA	XX10

Notes:
- Addresses in the table are the values in the CPU memory map. All addresses and data are hexadecimal values, where "x" is any value.
- RA: Read address
- PA: Program address. Only even addresses can be specified.
- RD: Read data
- PD: Program data. Only word data can be specified.

*: Two kinds of read/reset commands can reset flash memory to the read mode.

24.6 Confirming the Automatic Algorithm Execution State

Because the write/erase flow of the flash memory is controlled using the automatic algorithm, the flash memory has hardware for posting its internal operating state and completion of operation. This automatic algorithm enables confirmation of the operating state of the built-in flash memory using the following hardware sequences flag.

■ Hardware Sequence Flags

The hardware sequence flags are configured from the three-bit output of DQ7, DQ6, and DQ5. The functions of these bits are those of the data polling flag (DQ7), toggle bit flag (DQ6), and timing limit exceeded flag (DQ5). The hardware sequence flags can therefore be used to confirm that writing or chip erase has been completed or that erase code write is valid.

The hardware sequence flags can be accessed by read-accessing the addresses of the target sectors in the flash memory after setting of the command sequence (see Table 24.5-1). Table 24.6-1 lists the bit assignments of the hardware sequence flags.

Table 24.6-1 Bit Assignments of Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	–	–	–	–	–

To determine whether automatic writing or chip erase is being executed, the hardware sequence flags can be checked or the status can be determined from the RDY bit of the flash memory control status register (FMCS) that indicates whether writing has been completed. After writing/erasing has terminated, the state returns to the read/reset state. When creating a program, use one of the flags to confirm that automatic writing/erasing has terminated. Then, perform the next processing operation, such as data read. The following sections describe each hardware sequence flag separately. Table 24.6-2 lists the functions of the hardware sequence flags.

Table 24.6-2 Hardware Sequence Flag Function

State		DQ7	DQ6	DQ5
State change for normal operation	Write → Write completed (write address specified)	$\overline{\text{DQ7}}$ → DATA:7	Toggle → DATA:6	0 → DATA:5
	Chip/selector erase → Erase completed	0 → 1	Toggle → Stop	0 → 1
Abnormal operation	Write	$\overline{\text{DQ7}}$	Toggle	1
	Chip erase	0	Toggle	1

24.6.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) uses the data polling function to post that the automatic algorithm is being executed or has terminated

■ Data Polling Flag (DQ7)

Table 24.6-3 and Table 24.6-4 list the state transitions of the data polling flag.

Table 24.6-3 State Transition of Data Polling Flag (State change at normal operation)

Operating State	Programming → Completed	Chip Erasing → Completed
DQ7	$\overline{\text{DQ7}}$ → DATA:7	0 → 1

Table 24.6-4 State Transition of Data Polling Flag (State change at abnormal operation)

Operating State	Programming	Chip Erasing
DQ7	$\overline{\text{DQ7}}$	0

● Write

Read-access during execution of the automatic write algorithm causes the flash memory to output the opposite data of bit 7 last written, regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output bit 7 of the read value of the address specified by the address signal.

● Chip erase

Read-access during execution of the chip erase algorithm causes the flash memory to output 0. Read-access at the end of the automatic chip algorithm causes the flash memory to output 1 in the same way.

Note:

When the automatic algorithm is being started, read-access to the specified address is ignored. Since termination of the data polling flag (DQ7) can be accepted for a data read and other bits output, data read after the automatic algorithm has terminated should be performed after read-access has confirmed that data polling has terminated.

24.6.2 Toggle Bit Flag (DQ6)

Like the data polling flag (DQ7), the toggle bit flag (DQ6) uses the toggle bit function to post that the automatic algorithm is being executed or has terminated.

■ Toggle Bit Flag (DQ6)

Table 24.6-5 and Table 24.6-6 list the state transitions of the toggle bit flag.

Table 24.6-5 State Transition of Toggle Bit Flag (State change at normal operation)

Operating State	Programming → Completed	Chip Erasing → Completed
DQ6	Toggle → DATA:6	Toggle → Stop

Table 24.6-6 State Transition of Toggle Bit Flag (State change at abnormal operation)

Operating State	Programming	Chip Erasing
DQ6	Toggle	Toggle

● Write/chip erase

Continuous read-access during execution of the automatic write algorithm and chip erase algorithm causes the flash memory to toggle the 1 or 0 state for every read cycle, regardless of the value at the address specified by the address signal. Continuous read-access at the end of the automatic write algorithm and chip erase algorithm causes the flash memory to stop toggling bit 6 and output bit 6 (DATA: 6) of the read value of the address specified by the address signal.

24.6.3 Timing Limit Exceeded Flag (DQ5)

The timing limit exceeded flag (DQ5) is used to post that execution of the automatic algorithm has exceeded the time (internal pulse count) prescribed in the flash memory.

■ Timing Limit Exceeded Flag (DQ5)

Table 24.6-7 and Table 24.6-8 list the state transitions of the timing limit exceeded flag.

Table 24.6-7 State Transition of Timing Limit Exceeded Flag (State change at normal operation)

Operating State	Programming → Completed	Chip Erasing → Completed
DQ5	0 → DATA:5	0 → 1

Table 24.6-8 State Transition of Timing Limit Exceeded Flag (State change at abnormal operation)

Operating State	Programming	Chip Erasing
DQ5	1	1

● Write/chip erase

Read-access after write or chip erase automatic algorithm activation causes the flash memory to output 0 if the time is within the prescribed time (time required for write/erase) or to output 1 if the prescribed time has been exceeded. Because this is done regardless of whether the automatic algorithm is being executed or has terminated, it is possible to determine whether write/erase was successful or unsuccessful. That is, when this flag outputs 1, writing can be determined to have been unsuccessful if the automatic algorithm is still being executed by the data polling function or toggle bit function.

For example, writing 1 to a flash memory address where 0 has been written will cause the fail state to occur. In this case, the flash memory will lock and execution of the automatic algorithm will not terminate. In rare cases normal termination may be seen as with the case where "1" can be written. As a result, valid data will not be outputted from the data polling flag (DQ7). In addition, the toggle bit flag (DQ6) will exceed the time limit without stopping the toggle operation and the timing limit exceeded flag (DQ5) will output 1. Note that this state indicates that the flash memory is not faulty, but has not been used correctly. When this state occurs, execute the Reset command.

24.7 Detailed Explanation of Writing to and Erasing Flash Memory

This section describes each operation procedure of flash memory Read/Reset, Write, Chip Erase when a command that starts the automatic algorithm is issued.

■ Detailed Explanation of Flash Memory Write/erase

The flash memory executes the automatic algorithm by issuing a command sequence (see Table 24.5-1) for a write cycle to the bus to perform Read/Reset, Write, or Chip Erase operations. Each bus write cycle must be performed continuously. In addition, whether the automatic algorithm has terminated can be determined using the data polling or other function. At normal termination, the flash memory is returned to the read/ reset state.

Each operation of the flash memory is described in the following order:

- Setting the read/reset state
- Writing data
- Erasing all data (erasing chips)

24.7.1 Setting The Read/Reset State

This section describes the procedure for issuing the Read/Reset command to set the flash memory to the read/reset state.

■ Setting the Flash Memory to the Read/reset State

The flash memory can be set to the read/reset state by sending the Read/Reset command in the command sequence table (see Table 24.5-1) continuously to the target sector in the flash memory.

The Read/Reset command has two types of command sequences that execute the first and third bus operations. However, there are no essential differences between these command sequences.

The read/reset state is the initial state of the flash memory. When the power is turned on and when a command terminates normally, the flash memory is set to the read/reset state. In the read/reset state, other commands wait for input.

In the read/reset state, data is read by regular read-access. As with the mask ROM, program access from the CPU is enabled. The Read/Reset command is not required to read data by a regular read. The Read/Reset command is mainly used to initialize the automatic algorithm in such cases as when a command does not terminate normally.

24.7.2 Writing Data

This section describes the procedure for issuing the Write command to write data to the flash memory.

■ Writing Data to the Flash Memory

The data write automatic algorithm of the flash memory can be started by sending the Write command in the command sequence table (see Table 24.5-1) continuously to the target sector in the flash memory. When data write to the target address is completed in the fourth cycle, the automatic algorithm and automatic write are started.

● Specifying addresses

Only even addresses can be specified as the write addresses specified in a write data cycle. Odd addresses cannot be written correctly. That is, writing to even addresses must be done in units of word data.

Writing can be done in any order of addresses. However, the Write command writes only data of one word for each execution.

● Notes on writing data

Writing cannot return data 0 to data 1. When data 1 is written to data 0, the data polling algorithm (DQ7) or toggle operation (DQ6) does not terminate and the flash memory elements are determined to be faulty. If the time prescribed for writing is thus exceeded, the timing limit exceeded flag (DQ5) is determined to be an error. Otherwise, the data is viewed as if dummy data 1 had been written. However, when data is read in the read/reset state, the data remains 0. Data 0 can be set to data 1 only by erase operations.

All commands are ignored during execution of the automatic write algorithm. If a hardware reset is started during writing, the data of the written addresses will be unpredictable.

■ Writing to the Flash Memory

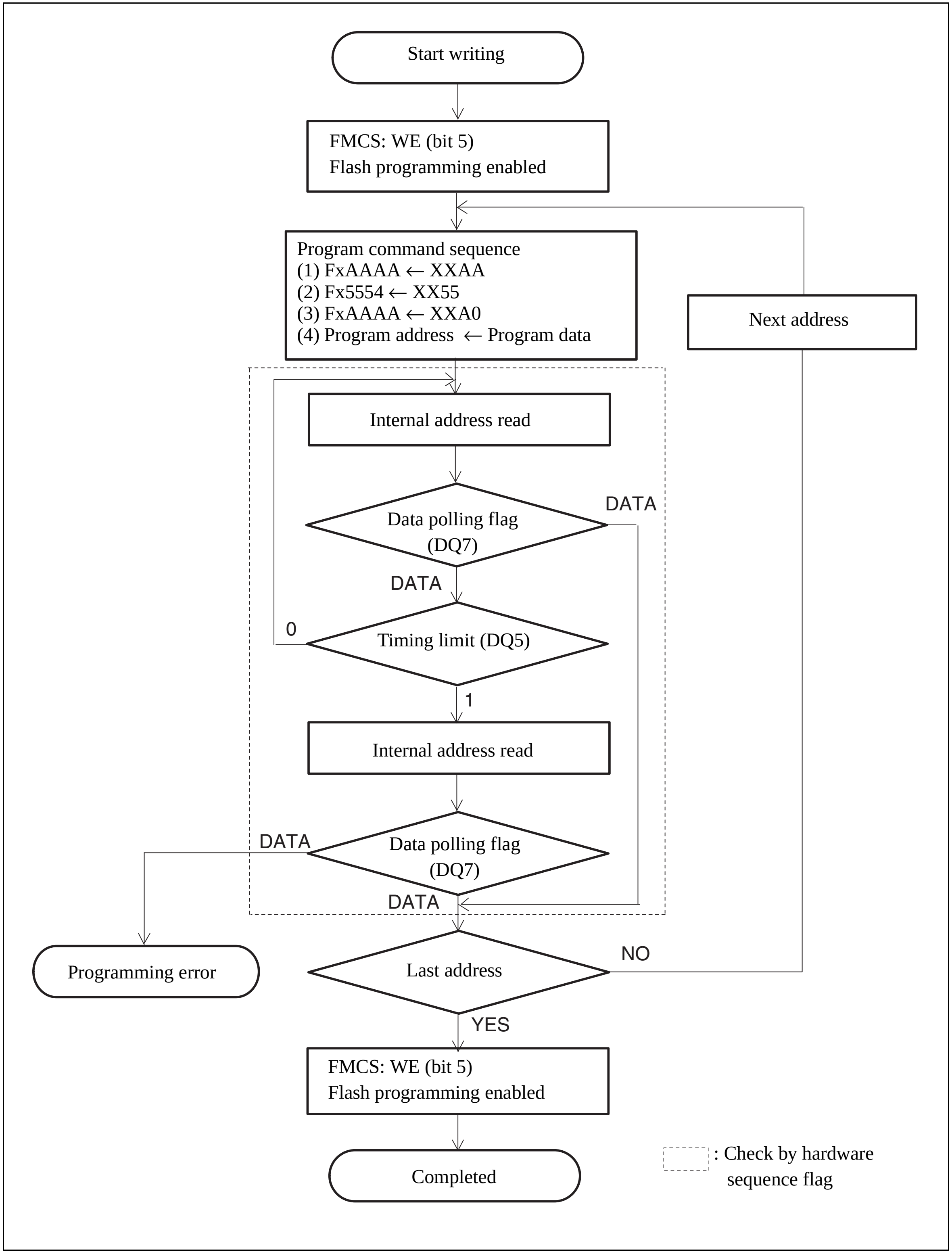
Figure 24.7-1 is an example of the procedure for writing to the flash memory. The hardware sequence flags (see "24.6 Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Here, the data polling flag (DQ7) is used to confirm that writing has terminated.

The data read to check the flag is read from the address written to last.

The data polling flag (DQ7) changes at the same time that the timing limit exceeded flag (DQ5) changes. For example, even if the timing limit exceeded flag (DQ5) is 1, the data polling flag bit (DQ7) must be rechecked.

Also for the toggle bit flag (DQ6), the toggle operation stops at the same time that the timing limit exceeded flag bit (DQ5) changes to 1. The toggle bit flag (DQ6) must therefore be rechecked.

Figure 24.7-1 Example of the Flash Memory Write Procedure



24.7.3 Erasing All Data (Erasing Chips)

This section describes the procedure for issuing the Chip Erase command to erase all data in the flash memory.

■ Erasing all Data in the Flash Memory (Erasing chips)

All data can be erased from the flash memory by sending the Chip Erase command in the command sequence table (see Table 24.5-1) continuously to the target sector in the flash memory.

The Chip Erase command is executed in six bus operations. When writing of the sixth cycle is completed, the chip erase operation is started. For chip erase, the user need not write to the flash memory before erasing. During execution of the automatic erase algorithm, the flash memory writes 0 for verification before all of the cells are erased automatically.

■ Erasing Chip in the Flash Memory

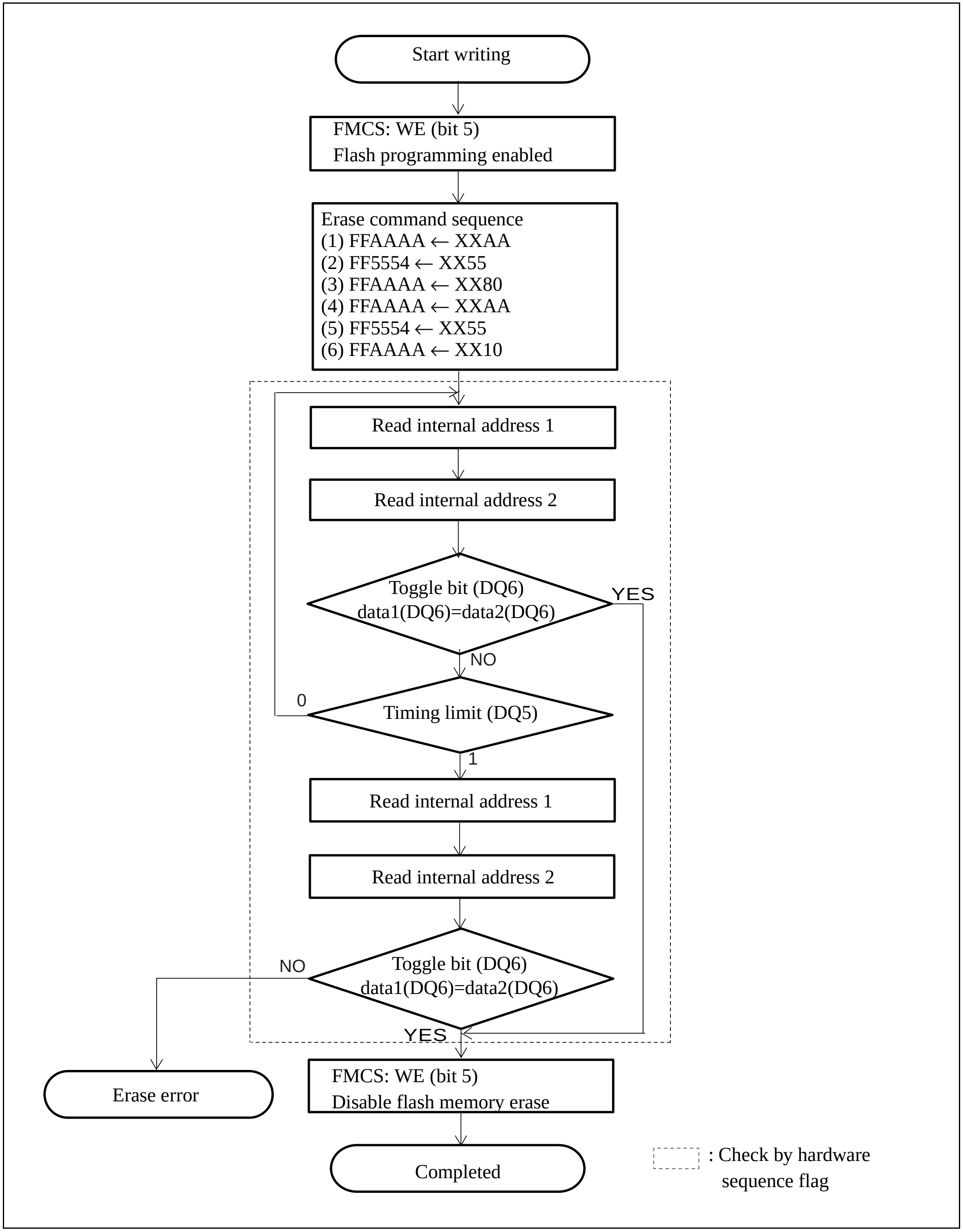
The hardware sequence flags (see "24.6 Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 24.7-2 is an example of the procedure for erasing chip in the flash memory. Here, the toggle bit flag (DQ6) is used to confirm that erasing has terminated.

The data that is read to check the flag is read from the sector to be erased.

The toggle bit flag (DQ6) stops the toggle operation at the same time that the timing limit exceeded flag (DQ5) is changed to 1. For example, even if the timing limit exceeded flag (DQ5) is 1, the toggle bit flag (DQ6) must be rechecked.

The data polling flag (DQ7) also changes at the same time that the timing limit exceeded flag bit (DQ5) changes. As a result, the data polling flag (DQ7) must be rechecked.

Figure 24.7-2 Example of the Flash Memory Chip Procedure



24.8 Notes on Using 512K-bit Flash Memory

This section contains notes on using 512K-bit flash memory.

■ Notes on Using Flash Memory

● Input of a hardware reset ($\overline{\text{RST}}$)

To input a hardware reset when the automatic algorithm has not been started and reading is in progress, a minimum "L" level width of 500 ns must be maintained. In this case, a maximum of 500 ns is required until data can be read from the flash memory after a hardware reset has been activated.

Similarly, to input a hardware reset when the automatic algorithm has been activated and writing or erasing is in progress, a minimum "L" level width of 500 ns must be maintained. In this case, 20 µs are required until data can be read after the operation for initializing the flash memory has terminated.

When a hardware reset is performed during writing, the data being written is undefined.

● Canceling of a software reset and watchdog timer reset

When the flash memory is being written to or erased with CPU access and if reset conditions occur while the automatic algorithm is active, the CPU may run out of control. This occurs because these reset conditions cause the automatic algorithm to continue without initializing the flash memory unit, possibly preventing the flash memory unit from entering the read state when the CPU starts the sequence after the reset has been deasserted. These reset conditions must be disabled during writing to or erasing of the flash memory.

● Program access to flash memory

When the automatic algorithm is operating, read access to the flash memory is disabled. With the memory access mode of the CPU set to internal ROM mode, writing or erasing must be started after the program area is switched to another area such as RAM.

● Extended intelligent I/O service (EI^{2}OS)

Because write and erase interrupts issued to the CPU from the flash memory interface circuit cannot be accepted by the EI^{2}OS, they should not be used.

24.9 Flash Security Feature

Flash security feature provides possibilities to protect the content of the flash memory.

■ Abstract

By writing the protection code of "01_H" to the security bit in the flash memory, access to the flash memory is restricted. Once the flash memory is protected, performing the chip erase operation only can unlock the function. Otherwise, read/write access to the flash memory from the external pins is not possible.

This function is suitable for applications requiring security of self-containing and data stored in the flash memory.

Address of the security bit depends on the size of built-in flash memory. Table 24.9-1 shows the address of security bit.

Table 24.9-1 Address of Flash Security Bit

	Flash memory size	Address of security bit
MB90F362/T(S), MB90F367/T(S)	Built-in 512K-bit flash memory	$FF0001_H$

■ How to Enable the Flash Security Feature

After writing the protection code "01_H" to the security bit, the following external reset or power-on enables the Flash Security Feature.

■ How to Disable the Flash Security Feature

Performing the chip erase operation.

■ Behavior Under the Flash Security Feature

Read operation: invalid data read

Write operation: ignored

■ Others

(1) About configuration of the general-purpose parallel programmer, please follow to the specification of parallel programmer.

(2) Writing the protection code at the last of flash memory programming is recommended, in order to prevent the device from enabling the Flash Security Feature accidentally.

Notes:

- The flash security bit is allocated in the flash memory area. When the protection code "01_H" is written to the security bit, it is locked by the security. So, when not using the security function, do not write "01_H" to this address.
  See Table 24.9-1 for the address of the security bit.
- By specifying the sector of the flash memory, it cannot be locked the security per sector. It is the security function for all areas in the flash memory.
- The FLASH memory trouble in the state to put security cannot be analyzed at all.

CHAPTER 25

EXAMPLES OF MB90F362/T(S), MB90F367/T(S) SERIAL PROGRAMMING CONNECTION

This chapter shows an example of a serial programming connection using the AF220/AF210/AF120/AF110 Flash Micro-computer Programmer by Yokogawa Digital Computer Corporation when the AF220/AF210/AF120/AF110 flash serial microcontroller programer from Yokogawa Digital Computer Corporation is used.

25.1 Basic Configuration of Serial Programming Connection with MB90F362/T(S), MB90F367/T(S)

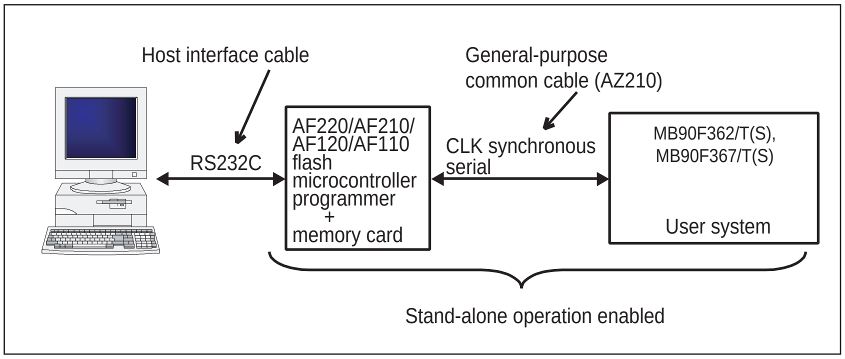
The MB90F362/T(S), MB90F367/T(S) supports on-board writing (Fujitsu standard) of the flash ROM. This section provides the related specifications.

■ Basic Configuration of Serial Programming Connection with MB90F362/T(S), MB90F367/T(S)

Fujitsu standard serial on-board writing uses the Yokogawa Digital Computer Corporation AF220/AF210/AF120/AF110 flash microcontroller programmer.

Figure 25.1-1 shows the basic configuration for the example serial programming connection of MB90F362/T(S), MB90F367/T(S).

Figure 25.1-1 Basic Configuration MB90F362/T(S), MB90F367/T(S) Serial Programming Connection


Host interface cable
General-purpose common cable (AZ210)
RS232C
AF220/AF210/ AF120/AF110 flash microcontroller programmer + memory card
CLK synchronous serial
MB90F362/T(S), MB90F367/T(S)
User system
Stand-alone operation enabled

Note:

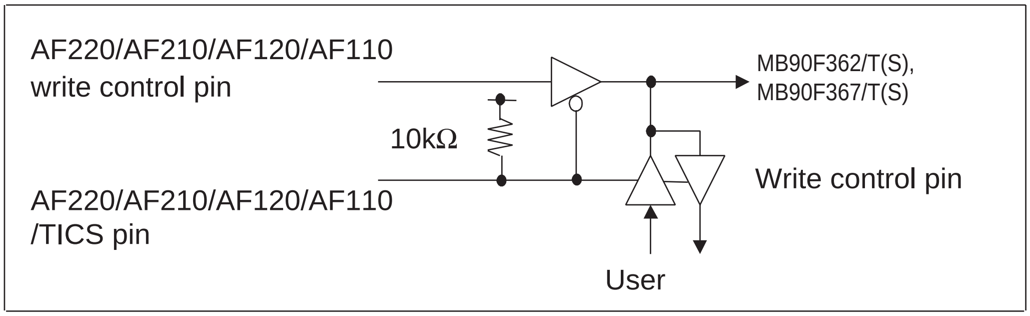
For information on the functions of and operational procedures related to the flash microcontroller programmer (AF220/AF210/AF120/AF110), the general-purpose common cable (AZ210) for connection, and the connector, contact Yokogawa Digital Computer Corporation.

Table 25.1-1 Pin Used for Fujitsu Standard Serial on-board Programming

Pin	Function	Additional information
MD2, MD1, MD0	Mode pins	Controls programming mode from the flash microcontroller programmer.
X0, X1	Oscillation pins	In programming mode, the CPU internal operation clock signal is one multiple of the PLL clock signal frequency. Therefore, because the oscillation clock frequency becomes the internal operation clock signal, the oscillator used for serial reprogramming is 4 MHz to 16 MHz.
P83, P84	Programming activation pins	Input "L" level to P83 and "H" level to P84.
$\overline{\text{RST}}$	Reset pin	-
SIN1	Serial data input pin	Use UART1 as CLK synchronous mode.
SOT1	Serial data output pin	
SCK1	Serial clock signal input pin	
C	C pin	This pin is used to stabilize the power supply. Connect to a external ceramic capacitor of approximately 0.1 µF or more.
V_{CC}	Power voltage supply pin	If the programming voltage (5 V ± 10%) is supplied from the user system, the flash microcontroller programmer need not be connected. Connect so that the power supply of the user side is not short-circuited.
V_{SS}	GND pin	Common to the ground of the flash microcontroller programmer.

Even if the P83, P84, SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in Figure 25.1-2 is required. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25.1-2 Control Circuit



"25.2 Example of Serial Programming Connection (User Power Supply Used)" to "25.5 Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)" present examples the following 4 types of serial programming connection. See each Section as required.

- Example of serial programming connection (user power supply used)
- Example of serial programming connection (power supplied from the programmer)
- Example of minimum connection to the flash microcontroller programmer (user power supply used)
- Example of minimum connection to the flash microcontroller programmer (power supplied from the

programmer)

Table 25.1-2 System Configuration of Flash Microcontroller Programmers (Manufactured by Yokogawa Digital Computer Corporation)

Model		Function
Main unit	AF220/AC4P	Ethernet interface built-in model and 100 V to 220 V AC power adapter
	AF210/AC4P	Standard model and 100 V to 220 V AC power adapter
	AF120/AC4P	Single-key Ethernet interface built-in model and 100 V to 220 V AC power adapter
	AF110/AC4P	Single-key model and 100 V to 220 V AC power adapter
AZ221		RS232C cable for programmer PC/AT
AZ210		Standard target probe (a) cable length: 1 m
FF201		Fujitsu F^{2}MC-16LX flash microcontroller control module
AZ290		Remote controller
/P2		2MB PC Card (optional) for flash memory sizes up to 128 KB
/P4		4MB PC Card (optional) for flash memory sizes up to 512 KB

Inquiries: Yokogawa Digital Computer Corporation
Telephone number: (81)-42-333-6224

Note:

Although the AF200 flash microcontroller programmer is no longer manufactured, the programmer still can be used in combination with the FF201 control module.

Examples of serial programming connection can correspond to the connection example as shown in "■Oscillating Clock Frequency and Serial Clock Input Frequency".

■ Oscillating Clock Frequency and Serial Clock Input Frequency

The equation listed below can be used to calculate the serial clock frequencies that can be used for the MB90F362/T(S), MB90F367/T(S).

Serial clock frequency that can be input = 0.125 * example of oscillation clock frequency

Set an appropriate serial clock input frequency in the flash microcontroller programmer according to the oscillating clock frequency in use.

Table 25.1-3 Examples of Serial Clock Frequencies that can be Input

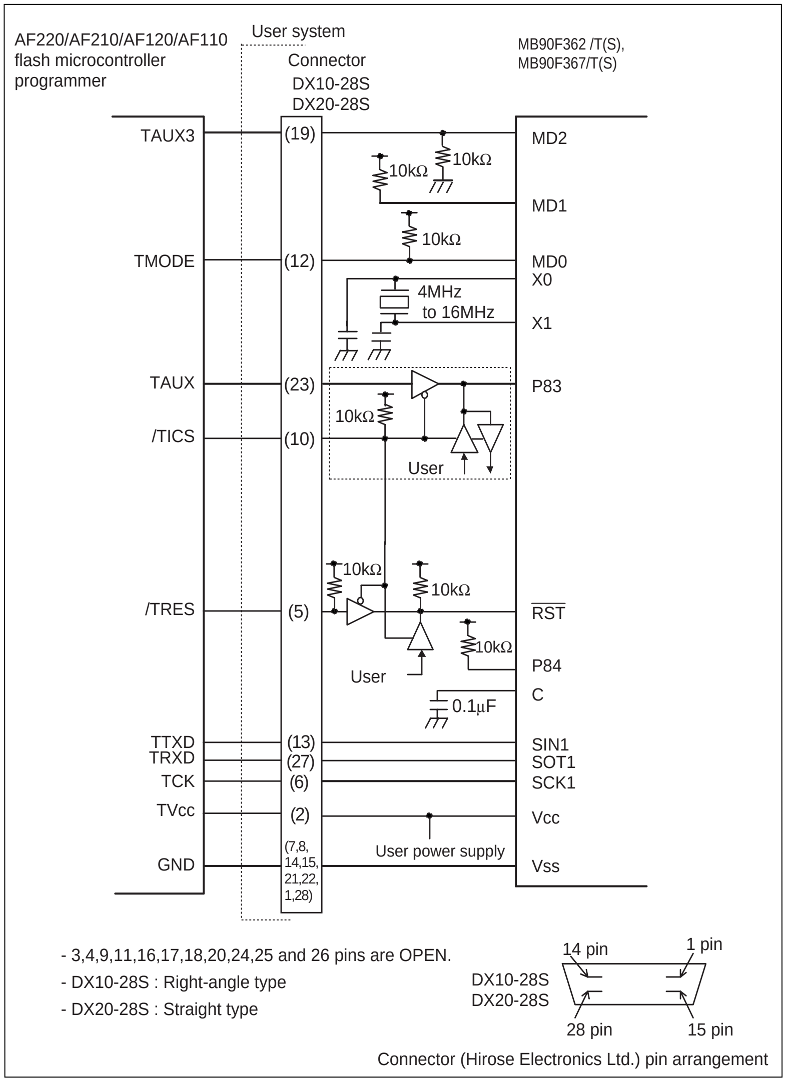
Oscillating clock frequency	Maximum serial clock frequency that can be input for microcontrollers	Maximum serial clock frequency that can be used for the AF220, AF210, AF120 and AF110	Maximum serial clock frequency that can be used for the AF200
4 MHz	500 kHz	500 kHz	500 kHz
8 MHz	1 MHz	850 kHz	500 kHz
16 MHz	2 MHz	1.25 MHz	500 kHz

25.2 Example of Serial Programming Connection (User Power Supply Used)

Figure 25.2-1 shows an example of a serial programming connection when the user power supply is used. The value 1 and 0 are input to mode pins MD2 and MD0 from TAUX3 and TMODE of the AF220/AF210/AF120/AF110 programmer.
Serial reprogramming mode: MD2, MD1, MD0 = 110.

■ Example of Serial Programming Connection (User power supply used)

Figure 25.2-1 Example of Serial Programming Connection for MB90F362/T(S), MB90F367/T(S) Single-chip Modes (User power supply Used)

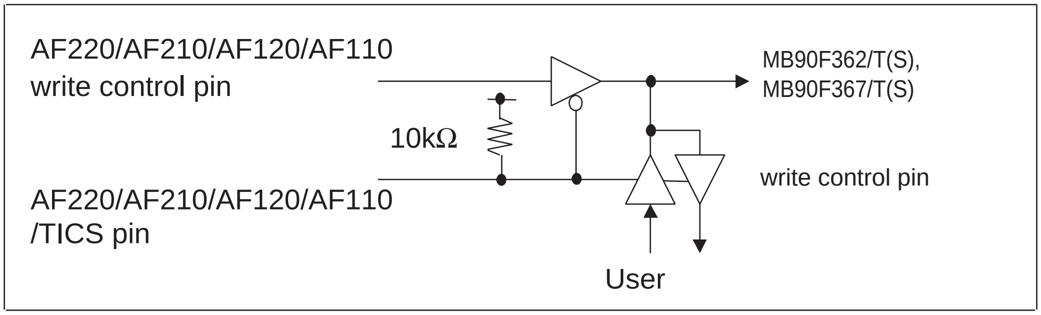


- 3,4,9,11,16,17,18,20,24,25 and 26 pins are OPEN.
- DX10-28S : Right-angle type
- DX20-28S : Straight type

Connector (Hirose Electronics Ltd.) pin arrangement

- Even if the SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in the figure below is required in the same way that it is for P83. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25.2-2 Control Circuit



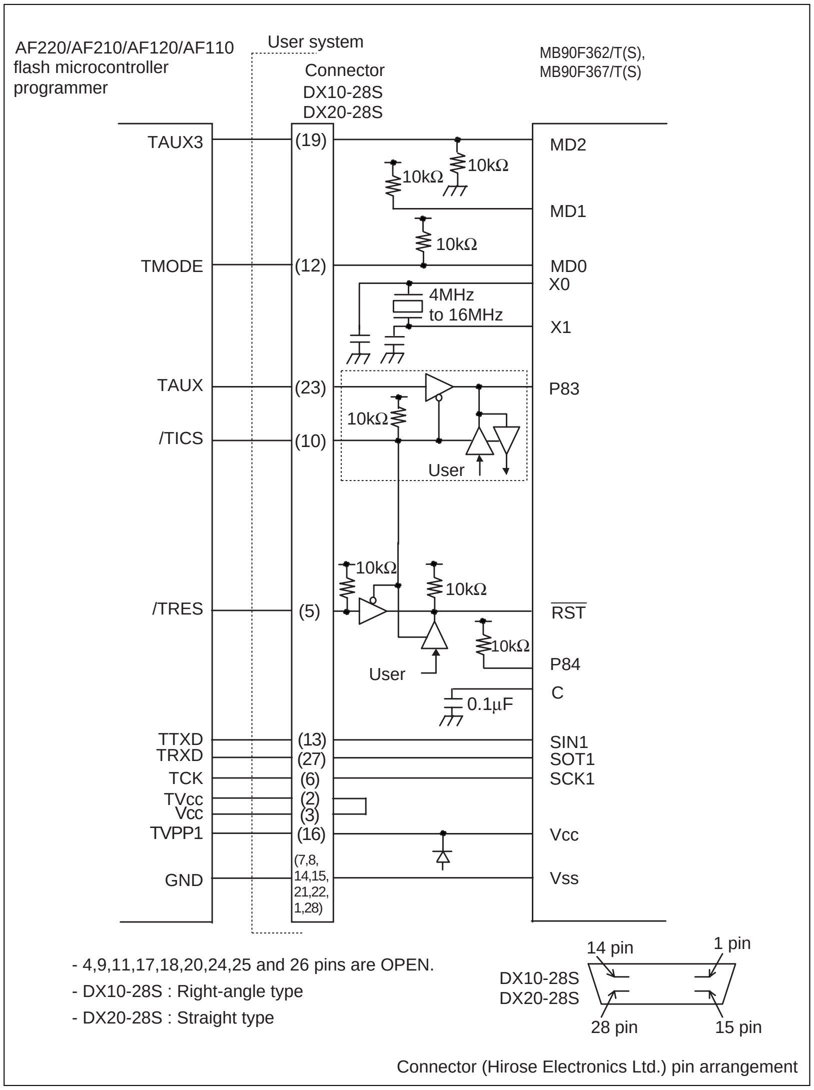
- Connect the AF220/AF210/AF120/AF110 while the user power is off.

25.3 Example of Serial Programming Connection (Power Supplied from Programmer)

Figure 25.3-1 shows an example of a serial programming connection when power is supplied from the programmer. The value 1 and 0 are input to mode pins MD2 and MD0 from TAUX3 and TMODE of the AF220/AF210/AF120/AF110 programmer.
Serial reprogramming mode: MD2, MD1, MD0 = 110.

■ Example of Serial Programming Connection (Power supplied from programmer)

Figure 25.3-1 Example of Serial Programming Connection for MB90F362/T(S), MB90F367/T(S) Single-chip Modes (Power supplied from programmer)

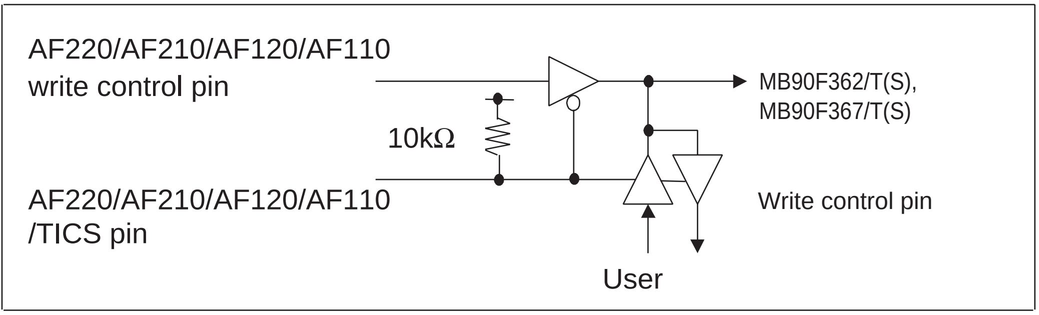


- 4,9,11,17,18,20,24,25 and 26 pins are OPEN.
- DX10-28S : Right-angle type
- DX20-28S : Straight type

Connector (Hirose Electronics Ltd.) pin arrangement

- Even if the SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in the figure below is required in the same way that it is for P83. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25.3-2 Control Circuit



- Connect the AF220/AF210/AF120/AF110 while the user power is off.
- When the programming power is supplied from the AF220/AF210/AF120/AF110, be careful not to short-circuit the user power supply.

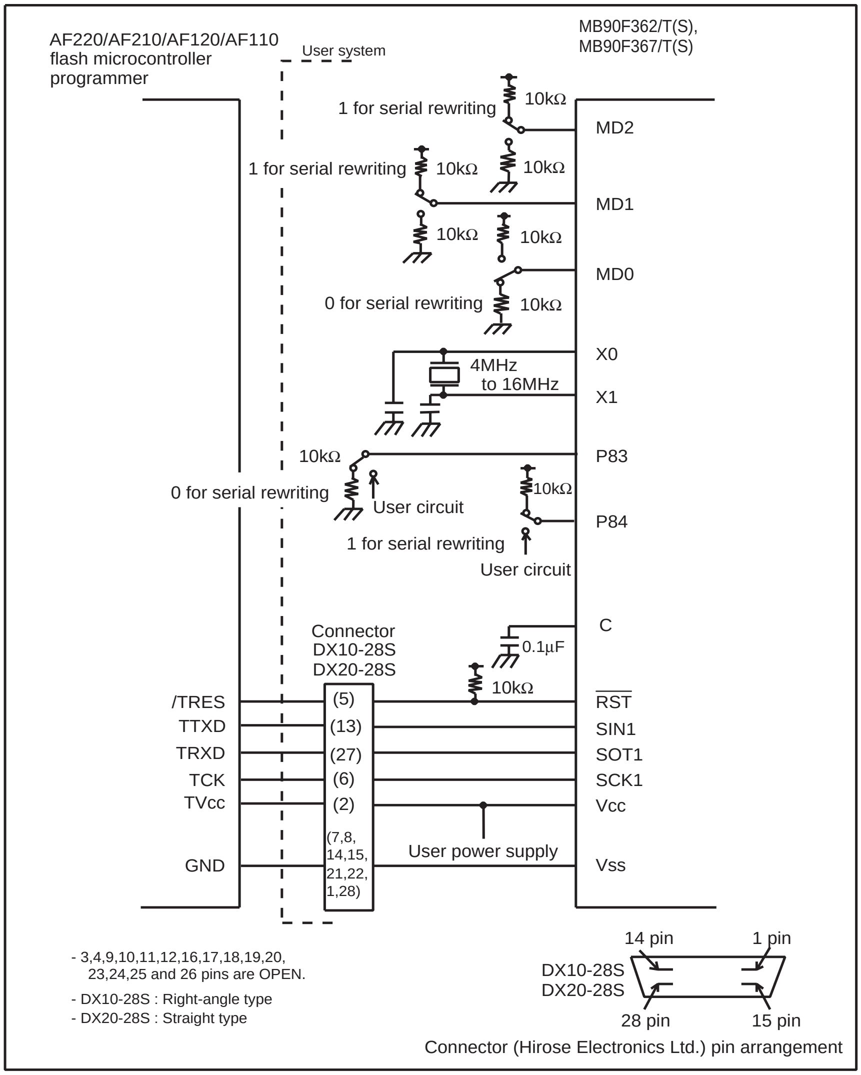
25.4 Example of Minimum Connection to Flash Microcontroller Programmer (User Power Supply Used)

Figure 25.4-1 shows an example of the minimum connection to the flash microcontroller programmer when the user power supply is used.
Serial reprogramming mode: MD2, MD1, MD0 = 110.

■ Example of Minimum Connection to Flash Microcontroller Programmer (User power supply used)

For a flash memory write, the MD2, MD1, MD0, P83, and P84 pins and flash microcontroller programmer need not be connected if the pins are set as Figure 25.4-1 .

Figure 25.4-1 Example of Minimum Connection to MB90F362/T(S), MB90F367/T(S) Flash Microcontroller Programmer (User power supply Used)

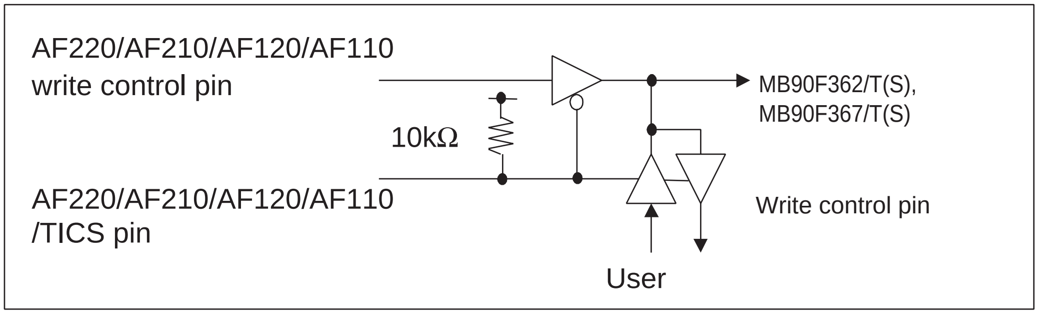


- 3,4,9,10,11,12,16,17,18,19,20, 23,24,25 and 26 pins are OPEN.
- DX10-28S : Right-angle type
- DX20-28S : Straight type

Connector (Hirose Electronics Ltd.) pin arrangement

- Even if the SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in the figure below is required. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25.4-2 Control Circuit



- Connect the AF220/AF210/AF120/AF110 while the user power is off.

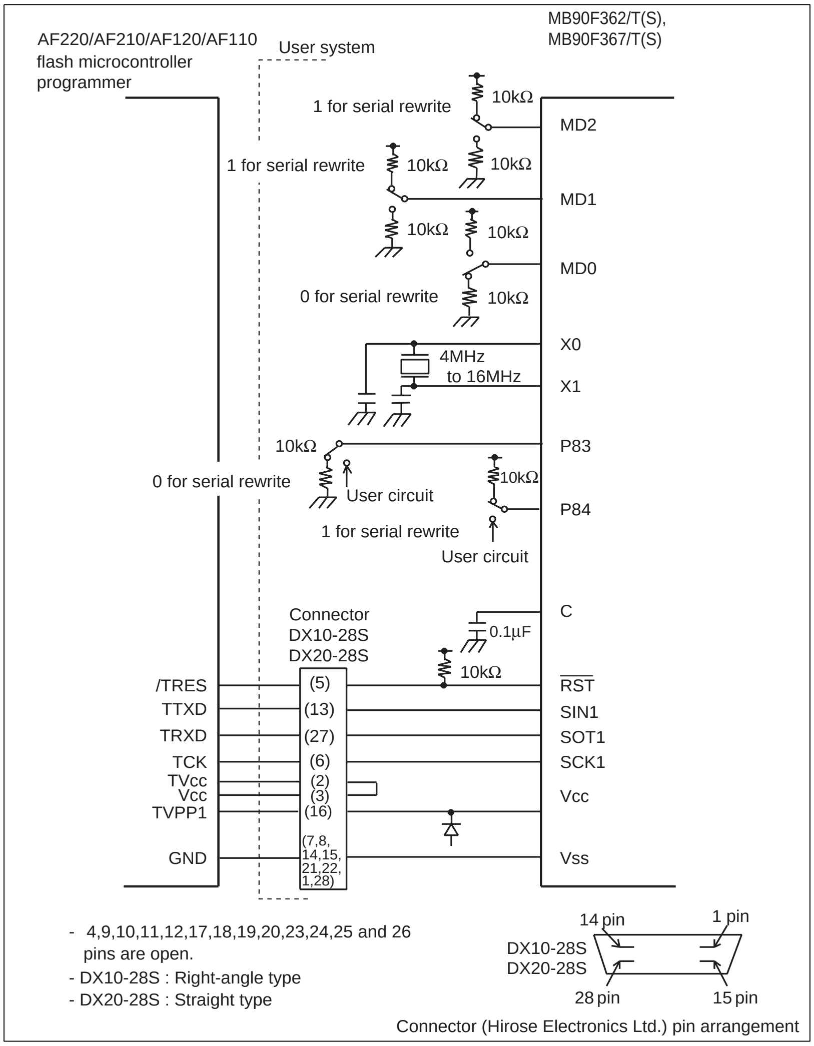
25.5 Example of Minimum Connection to Flash Microcontroller Programmer (Power Supplied from Programmer)

Figure 25.5-1 shows an example of the minimum connection to the MB90F362/T(S), MB90F367/T(S) flash microcontroller programmer when power is supplied from the Programmer.
Serial reprogramming mode: MD2, MD1, MD0 = 110$_B$.

■ Example of Minimum Connection to Flash Microcomputer Programmer (Power supplied from programmer)

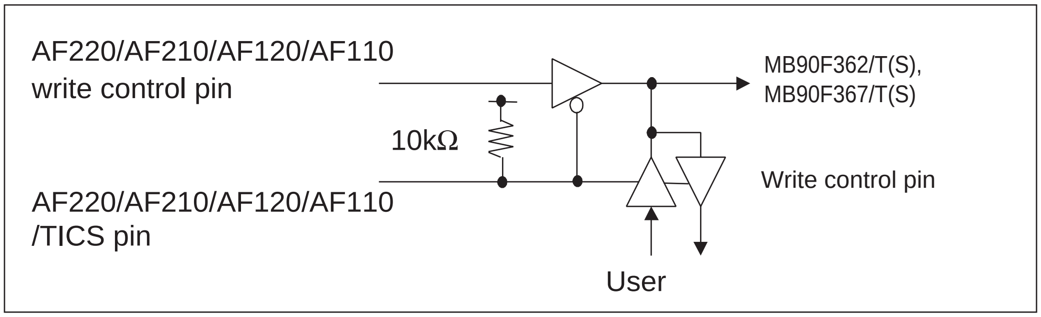
For a flash memory write, the MD2, MD1, MD0, P83, and P84 pins and flash microcontroller programmer need not be connected if the pins are set as Figure 25.5-1 .

Figure 25.5-1 Example of Minimum Connection to the MB90F362/T(S), MB90F367/T(S) Flash Microcontroller Programmer (power supplied from programmer)



- Even if the SIN1, SOT1, and SCK1 pins are used for the user system, the control circuit shown in the figure below is required. (The /TICS signal of the flash microcontroller programmer can be used to disconnect the user circuit during serial programming.)

Figure 25.5-2 Control Circuit



- Connect the AF220/AF210/AF120/AF110 while the user power is off.
- When the programming power is supplied from the AF220/AF210/AF120/AF110, be careful not to short-circuit the user power supply.

CHAPTER 26

ROM SECURITY FUNCTION

This chapter explains the ROM security function.

26.1 Overview of ROM Security Function

The ROM security function protects the content of ROM.

■ Overview of ROM Security Function

The ROM security function is a function to prevent ROM data being read to the third party by limiting the access to ROM.

Please contact to Fujitsu about details of this function.

APPENDIX

The appendixes provide I/O maps, instructions, and other information.

APPENDIX A I/O Maps

Table A-1 lists addresses to be assigned to the registers in the peripheral blocks.

■ I/O Maps (00XX Addresses)

Table A-1 I/O Map (1/5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
000000_H to 000001_H	Reserved				
000002_H	Port 2 data register	PDR2	R/W	Port 2	$XXXXXXXX_B$
000003_H	Reserved				
000004_H	Port 4 data register	PDR4	R/W	Port 4	$XXXXXXXX_B$
000005_H	Port 5 data register	PDR5	R/W	Port 5	$XXXXXXXX_B$
000006_H	Port 6 data register	PDR6	R/W	Port 6	$XXXXXXXX_B$
000007_H	Reserved				
000008_H	Port 8 data register	PDR8	R/W	Port 8	$XXXXXXXX_B$
000009_H to $00000A_H$	Reserved				
$00000B_H$	Analog input enable port 5	ADER5	R/W	Port 5, A/D	1 1 1 1 1 1 1 1_B
$00000C_H$	Analog input enable port 6	ADER6	R/W	Port 6, A/D	1 1 1 1 1 1 1 1_B
$00000D_H$	Reserved				
$00000E_H$	Input level select register0	ILSR0	R/W	Ports	$XXXXXXXX_B$
$00000F_H$	Input level select register1	ILSR1	R/W	Ports	$XXXXXXXX_B$
000010_H	Reserved				
000011_H					
000012_H	Port 2 direction register	DDR2	R/W	Port 2	0 0 0 0 0 0 0 0_B
000013_H	Reserved				
000014_H	Port 4 direction register	DDR4	R/W	Port 4	XXX 0 0 0 0 0_B
000015_H	Port 5 direction register	DDR5	R/W	Port 5	0 0 0 0 0 0 0 0_B
000016_H	Port 6 direction register	DDR6	R/W	Port 6	0 0 0 0 0 0 0 0_B
000017_H	Reserved				
000018_H	Port 8 direction register	DDR8	R/W	Port 8	0 0 0 0 0 0 X 0_B

Table A-1 I/O Map (2/5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
000019$_H$	Reserved				
00001A$_H$	Port A direction register	DDRA	W	Port A	XXX0 0XXX$_B$
00001B$_H$ to 00001D$_H$	Reserved				
00001E$_H$	Port 2 pull-up control register	PUCR2	R/W	Port 2	0 0 0 0 0 0 0 0$_B$
00001F$_H$	Reserved				
000020$_H$	Serial mode register 0	SMR0	W, R/W	UART0	0 0 0 0 0 0 0 0$_B$
000021$_H$	Serial control register 0	SCR0	W, R/W		0 0 0 0 0 0 0 0$_B$
000022$_H$	Reception/transmission data register 0	RDR0/TDR0	R/W		0 0 0 0 0 0 0 0$_B$
000023$_H$	Serial status register 0	SSR0	R, R/W		0 0 0 0 1 0 0 0$_B$
000024$_H$	Extended communication control register 0	ECCR0	R, W, R/W		0 0 0 0 0 0 XX$_B$
000025$_H$	Extended status control register0	ESCR0	R/W		0 0 0 0 0 1 0 0$_B$
000026$_H$	Baud rate generator register 00	BGR00	R/W, R		0 0 0 0 0 0 0 0$_B$
000027$_H$	Baud rate generator register 01	BGR01	R/W, R		0 0 0 0 0 0 0 0$_B$
000028$_H$	Serial mode register 1	SMR1	W, R/W	UART1	0 0 0 0 0 0 0 0$_B$
000029$_H$	Serial control register 1	SCR1	W, R/W		0 0 0 0 0 0 0 0$_B$
00002A$_H$	Reception/transmission data register 1	RDR1/TDR1	R/W		0 0 0 0 0 0 0 0$_B$
00002B$_H$	Serial status register 1	SSR1	R, R/W		0 0 0 0 1 0 0 0$_B$
00002C$_H$	Extended communication control register1	ECCR1	R, W, R/W		0 0 0 0 0 0 XX$_B$
00002D$_H$	Extended status control register 1	ESCR1	R/W		0 0 0 0 0 1 0 0$_B$
00002E$_H$	Baud rate generator register 10	BGR10	R/W, R		0 0 0 0 0 0 0 0$_B$
00002F$_H$	Baud rate generator register 11	BGR11	R/W, R		0 0 0 0 0 0 0 0$_B$
000030$_H$ to 00003A$_H$	Reserved				
00003B$_H$	Address detection control register 1	PACSR1	R/W	Address Match Detection 1	0 0 0 0 0 0 0 0$_B$
00003C$_H$ to 000047$_H$	Reserved				
000048$_H$	PPGC operation mode control register	PPGCC	W, R/W	16-bit PPGC/D	0 X 0 0 0 XX1$_B$
000049$_H$	PPGD operation mode control register	PPGCD	W, R/W		0 X 0 0 0 0 0 1$_B$
00004A$_H$	PPGC /D count clock selection register	PPGCD	R/W		0 0 0 0 0 0 X 0$_B$

Table A-1 I/O Map (3/5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
00004B$_H$	Reserved				
00004C$_H$	PPGE operation mode control register	PPGCE	W, R/W	16-bit PPGE/F	0 X 0 0 0 XX1$_B$
00004D$_H$	PPGF operation mode control register	PPGCF	W, R/W		0 X 0 0 0 0 0 1$_B$
00004E$_H$	PPGE/F count clock selection register	PPGEF	R/W		0 0 0 0 0 0 X 0$_B$
00004F$_H$	Reserved				
000050$_H$	Input capture control status 0/1	ICS01	R/W	Input Capture 0/1	0 0 0 0 0 0 0 0$_B$
000051$_H$	Input capture edge 0/1	ICE01	R/W, R		XXX0X0XX$_B$
000052$_H$	Input capture control status 2/3	ICS23	R/W	Input Capture 2/3	0 0 0 0 0 0 0 0$_B$
000053$_H$	Input capture edge 2/3	ICE23	R		XXXXXXXX$_B$
000054$_H$ to 000063$_H$	Reserved				
000064$_H$	Timer control status 2	TMCSR2	R/W	16-bit Reload Timer 2	0 0 0 0 0 0 0 0$_B$
000065$_H$	Timer control status 2	TMCSR2	R/W		XXXX 0 0 0 0$_B$
000066$_H$	Timer control status 3	TMCSR3	R/W	16-bit Reload Timer 3	0 0 0 0 0 0 0 0$_B$
000067$_H$	Timer control status 3	TMCSR3	R/W		XXXX 0 0 0 0$_B$
000068$_H$	A/D control status 0	ADCS0	R/W	A/D Converter	0 0 0 XXXX 0$_B$
000069$_H$	A/D control status 1	ADCS1	R/W, W		0 0 0 0 0 0 0 X$_B$
00006A$_H$	A/D data 0	ADCR0	R		0 0 0 0 0 0 0 0$_B$
00006B$_H$	A/D data 1	ADCR1	R		XXXXXX 0 0$_B$
00006C$_H$	ADC setting 0	ADSR0	R/W		0 0 0 0 0 0 0 0$_B$
00006D$_H$	ADC setting 1	ADSR1	R/W		0 0 0 0 0 0 0 0$_B$
00006E$_H$	Detection reset control register of low-voltage/CPU operation	LVRC	R/W, W	Detection Reset of Low-voltage/CPU Operation	0 0 1 1 1 0 0 0$_B$
00006F$_H$	ROM mirror function select	ROMM	W	ROM Mirror	XXXXXXX1$_B$
000070$_H$ to 00007F$_H$	Reserved				
000080$_H$ to 00008F$_H$	Reserved for CAN interface. (For more information, see Table 21.3-1 .)				
000090$_H$ to 00009D$_H$	Reserved				
00009E$_H$	Address detection control register0	PACSR0	R/W	Address Match Detection 0	0 0 0 0 0 0 0 0$_B$

Table A-1 I/O Map (4/5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
00009F$_H$	Delayed interrupt/release register	DIRR	R/W	Delayed Interrupt Generation Module	XXXXXXX0$_B$
0000A0$_H$	Low-power mode control register	LPMCR	W, R/W	Low Power Controller	0 0 0 1 1 0 0 0$_B$
0000A1$_H$	Clock selection register	CKSCR	R, R/W	Low Power Controller	1 1 1 1 1 1 0 0$_B$
0000A2$_H$ to 0000A7$_H$	Reserved				
0000A8$_H$	Watchdog timer control register	WDTC	R, W	Watchdog Timer	XXXXX111$_B$
0000A9$_H$	Time base timer control register	TBTC	W, R/W	Time Base Timer	1 XX 0 0 1 0 0$_B$
0000AA$_H$	Watch timer control register	WTC	R, R/W	Watch Timer	1 X 0 0 1 0 0 0$_B$
0000AB$_H$ to 0000AD$_H$	Reserved				
0000AE$_H$	Flash control status (Flash device only.)	FMCS	R, R/W	Flash Memory	0 0 0 X 0 0 0 0$_B$
0000AF$_H$	Reserved				
0000B0$_H$	Interrupt control register 00	ICR00	W, R/W	Interrupt Controller	0 0 0 0 0 1 1 1$_B$
0000B1$_H$	Interrupt control register 01	ICR01	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B2$_H$	Interrupt control register 02	ICR02	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B3$_H$	Interrupt control register 03	ICR03	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B4$_H$	Interrupt control register 04	ICR04	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B5$_H$	Interrupt control register 05	ICR05	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B6$_H$	Interrupt control register 06	ICR06	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B7$_H$	Interrupt control register 07	ICR07	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B8$_H$	Interrupt control register 08	ICR08	W, R/W		0 0 0 0 0 1 1 1$_B$
0000B9$_H$	Interrupt control register 09	ICR09	W, R/W		0 0 0 0 0 1 1 1$_B$
0000BA$_H$	Interrupt control register 10	ICR10	W, R/W		0 0 0 0 0 1 1 1$_B$
0000BB$_H$	Interrupt control register 11	ICR11	W, R/W		0 0 0 0 0 1 1 1$_B$
0000BC$_H$	Interrupt control register 12	ICR12	W, R/W		0 0 0 0 0 1 1 1$_B$
0000BD$_H$	Interrupt control register 13	ICR13	W, R/W		0 0 0 0 0 1 1 1$_B$
0000BE$_H$	Interrupt control register 14	ICR14	W, R/W		0 0 0 0 0 1 1 1$_B$
0000BF$_H$	Interrupt control register 15	ICR15	W, R/W		0 0 0 0 0 1 1 1$_B$
0000C0$_H$ to 0000C9$_H$	Reserved				

Table A-1 I/O Map (5/5)

Address	Register	Abbreviation	Access	Peripheral	Initial value
0000CA$_H$	External interrupt enable 1	ENIR1	R/W	External Interrupt 1	0 0 0 0 0 0 0 0$_B$
0000CB$_H$	External interrupt request 1	EIRR1	R/W		XXXXXXXX$_B$
0000CC$_H$	External interrupt level 1	ELVR1	R/W		0 0 0 0 0 0 0 0$_B$
0000CD$_H$	External interrupt level 1	ELVR1	R/W		0 0 0 0 0 0 0 0$_B$
0000CE$_H$	External interrupt 1 source select	EISSR	R/W		0 0 0 0 0 0 0 0$_B$
0000CF$_H$	PLL/subclock control register	PSCCR	W	PLL	XXXX 0 0 0 0$_B$
0000D0$_H$ to 0000FF$_H$	Reserved				

■ I/O map (79XX - 7FXX addresses)

Table A-2 I/O Map (7900$_H$ - 7FFF$_H$) (1/3)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7900$_H$ to 7917$_H$	Reserved				
7918$_H$	Reload register LC	PRLLC	R/W	16-bit PPG C/D	XXXXXXXX$_B$
7919$_H$	Reload register HC	PRLHC	R/W		XXXXXXXX$_B$
791A$_H$	Reload register LD	PRLLD	R/W		XXXXXXXX$_B$
791B$_H$	Reload register HD	PRLHD	R/W		XXXXXXXX$_B$
791C$_H$	Reload register LE	PRLLE	R/W	16-bit PPG E/F	XXXXXXXX$_B$
791D$_H$	Reload register HE	PRLHE	R/W		XXXXXXXX$_B$
791E$_H$	Reload register LF	PRLLF	R/W		XXXXXXXX$_B$
791F$_H$	Reload register HF	PRLHF	R/W		XXXXXXXX$_B$
7920$_H$	Input capture 0	IPCP0	R	Input Capture 0/1	XXXXXXXX$_B$
7921$_H$	Input capture 0	IPCP0	R		XXXXXXXX$_B$
7922$_H$	Input capture 1	IPCP1	R		XXXXXXXX$_B$
7923$_H$	Input capture 1	IPCP1	R		XXXXXXXX$_B$
7924$_H$	Input capture 2	IPCP2	R	Input Capture 2/3	XXXXXXXX$_B$
7925$_H$	Input capture 2	IPCP2	R		XXXXXXXX$_B$
7926$_H$	Input capture 3	IPCP3	R		XXXXXXXX$_B$
7927$_H$	Input capture 3	IPCP3	R		XXXXXXXX$_B$
7928$_H$ to 793F$_H$	Reserved				
7940$_H$	Timer data 0	TCDT0	R/W	I/O Timer 0	0 0 0 0 0 0 0 0$_B$
7941$_H$	Timer data 0	TCDT0	R/W		0 0 0 0 0 0 0 0$_B$
7942$_H$	Timer control 0	TCCSL0	R/W		0 0 0 0 0 0 0 0$_B$
7943$_H$	Timer control 0	TCCSH0	R/W		0XXXXXXX$_B$
7944$_H$ to 794B$_H$	Reserved				
794C$_H$	Timer 2/reload 2	TMR2/ TMRLR2	R, W	16-bit Reload Timer 2	XXXXXXXX$_B$
794D$_H$			R, W		XXXXXXXX$_B$
794E$_H$	Timer 3/reload 3	TMR3/ TMRLR3	R, W	16-bit Reload Timer 3	XXXXXXXX$_B$
794F$_H$			R, W		XXXXXXXX$_B$

Table A-2 I/O Map (7900_H - $7FFF_H$) (2/3)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7950_H to $795F_H$	Reserved				
7960_H	Clock supervisor control register	CSVCR	R, R/W	Clock supervisor	$0\,0\,0\,1\,1\,1\,0\,0_B$
7961_H to $796D_H$	Reserved				
$796E_H$	CAN direct mode register (For only MB90V340)	CDMR	R/W	CAN Clock Sync	XXXXXXX0B
$796F_H$ to $79DF_H$	Reserved				
$79E0_H$	Detection address setting 0	PADR0	R/W	Address Match Detection 0	$XXXXXXXX_B$
$79E1_H$	Detection address setting 0	PADR0	R/W		$XXXXXXXX_B$
$79E2_H$	Detection address setting 0	PADR0	R/W		$XXXXXXXX_B$
$79E3_H$	Detection address setting 1	PADR1	R/W		$XXXXXXXX_B$
$79E4_H$	Detection address setting 1	PADR1	R/W		$XXXXXXXX_B$
$79E5_H$	Detection address setting 1	PADR1	R/W		$XXXXXXXX_B$
$79E6_H$	Detection address setting 2	PADR2	R/W		$XXXXXXXX_B$
$79E7_H$	Detection address setting 2	PADR2	R/W		$XXXXXXXX_B$
$79E8_H$	Detection address setting 2	PADR2	R/W		$XXXXXXXX_B$
$79E9_H$ to $79EF_H$	Reserved				
$79F0_H$	Detection address setting 3	PADR3	R/W	Address Match Detection 1	$XXXXXXXX_B$
$79F1_H$	Detection address setting 3	PADR3	R/W		$XXXXXXXX_B$
$79F2_H$	Detection address setting 3	PADR3	R/W		$XXXXXXXX_B$
$79F3_H$	Detection address setting 4	PADR4	R/W		$XXXXXXXX_B$
$79F4_H$	Detection address setting 4	PADR4	R/W		$XXXXXXXX_B$
$79F5_H$	Detection address setting 4	PADR4	R/W		$XXXXXXXX_B$
$79F6_H$	Detection address setting 5	PADR5	R/W		$XXXXXXXX_B$
$79F7_H$	Detection address setting 5	PADR5	R/W		$XXXXXXXX_B$
$79F8_H$	Detection address setting 5	PADR5	R/W		$XXXXXXXX_B$
$79F9_H$ to $7BFF_H$	Reserved				
$7C00_H$ to $7CFF_H$	Reserved for CAN interface 1. (For more information, see Table 21.3-1 .)				

Table A-2 I/O Map (7900$_H$ - 7FFF$_H$) (3/3)

Address	Register	Abbreviation	Access	Peripheral	Initial value
7D00$_H$ to 7DFF$_H$	Reserved for CAN interface 1. (For more information, see Table 21.3-2 .)				
7E00$_H$ to 7FFF$_H$	Reserved				

Note:

Any write access to reserved addresses in I/O map should not be perfoermed.
A read access to reserved address results in reading "X".

- Explanation of write and read

R/W: Both read and write enabled

R: Only read enabled

W: Only write enabled

- Explanation of initial values

0: The initial value of this bit is "0".

1: The initial value of this bit is "1".

X: The initial value of this bit is undefined.

APPENDIX B Instructions

Appendix B describes the instructions used by the F^{2}MC-16LX.

B.1 Instruction Types

The F^{2}MC-16LX supports 351 types of instructions. Addressing is enabled by using an effective address field of each instruction or using the instruction code itself.

■ Instruction Types

The F^{2}MC-16LX supports the following 351 types of instructions:

- 41 transfer instructions (byte)
- 38 transfer instructions (word or long word)
- 42 addition/subtraction instructions (byte, word, or long word)
- 12 increment/decrement instructions (byte, word, or long word)
- 11 comparison instructions (byte, word, or long word)
- 11 unsigned multiplication/division instructions (word or long word)
- 11 signed multiplication/division instructions (word or long word)
- 39 logic instructions (byte or word)
- 6 logic instructions (long word)
- 6 sign inversion instructions (byte or word)
- 1 normalization instruction (long word)
- 18 shift instructions (byte, word, or long word)
- 50 branch instructions
- 6 accumulator operation instructions (byte or word)
- 28 other control instructions (byte, word, or long word)
- 21 bit operation instructions
- 10 string instructions

B.2 Addressing

With the F^{2}MC-16LX, the address format is determined by the instruction effective address field or the instruction code itself (implied). When the address format is determined by the instruction code itself, specify an address in accordance with the instruction code used. Some instructions permit the user to select several types of addressing.

■ Addressing

The F^{2}MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)

■ Effective Address Field

Table B.2-1 lists the address formats specified by the effective address field.

Table B.2-1 Effective Address Field

Code	Representation			Address format	Default bank
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	DTB
09	@RW1				DTB
0A	@RW2				ADB
0B	@RW3				SPB
0C	@RW0+			Register indirect with post increment	DTB
0D	@RW1+				DTB
0E	@RW2+				ADB
0F	@RW3+				SPB
10	@RW0+disp8			Register indirect with 8-bit displacement	DTB
11	@RW1+disp8				DTB
12	@RW2+disp8				ADB
13	@RW3+disp8				SPB
14	@RW4+disp8			Register indirect with 8-bit displacement	DTB
15	@RW5+disp8				DTB
16	@RW6+disp8				ADB
17	@RW7+disp8				SPB
18	@RW0+disp16			Register indirect with 16-bit displacement	DTB
19	@RW1+disp16				DTB
1A	@RW2+disp16				ADB
1B	@RW3+disp16				SPB
1C	@RW0+RW7			Register indirect with index	DTB
1D	@RW1+RW7			Register indirect with index	DTB
1E	@PC+disp16			PC indirect with 16-bit displacement	PCB
1F	addr16			Direct address	DTB

B.3 Direct Addressing

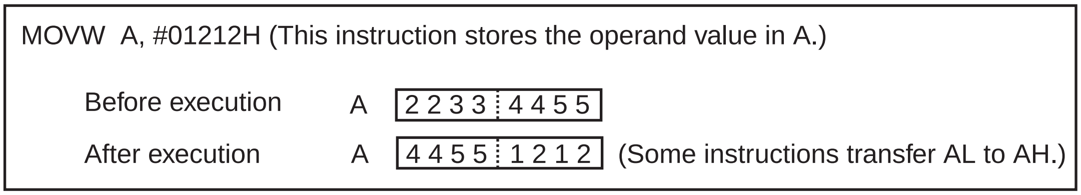
An operand value, register, or address is specified explicitly in direct addressing mode.

■ Direct Addressing

● Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

Figure B.3-1 Example of Immediate Addressing (#imm)


MOVW A, #01212H (This instruction stores the operand value in A.)
Before execution
A
2 2 3 3
4 4 5 5
After execution
A
4 4 5 5
1 2 1 2
(Some instructions transfer AL to AH.)

● Register direct addressing

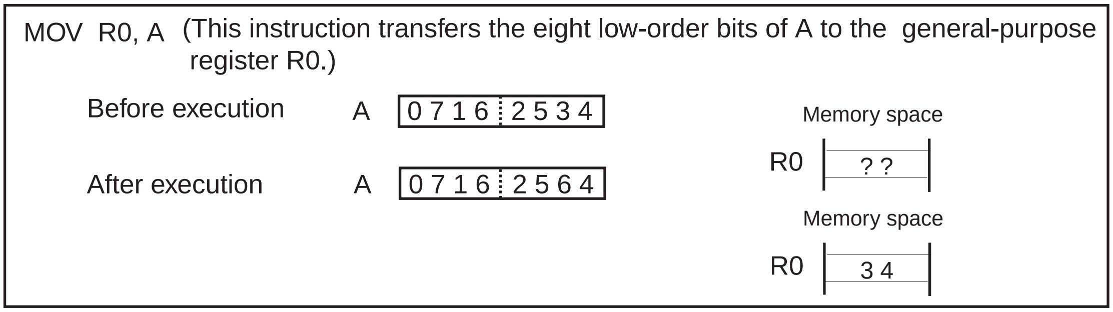
Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

Table B.3-1 Direct Addressing Registers

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, R5W, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP *
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*: One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified implicitly.

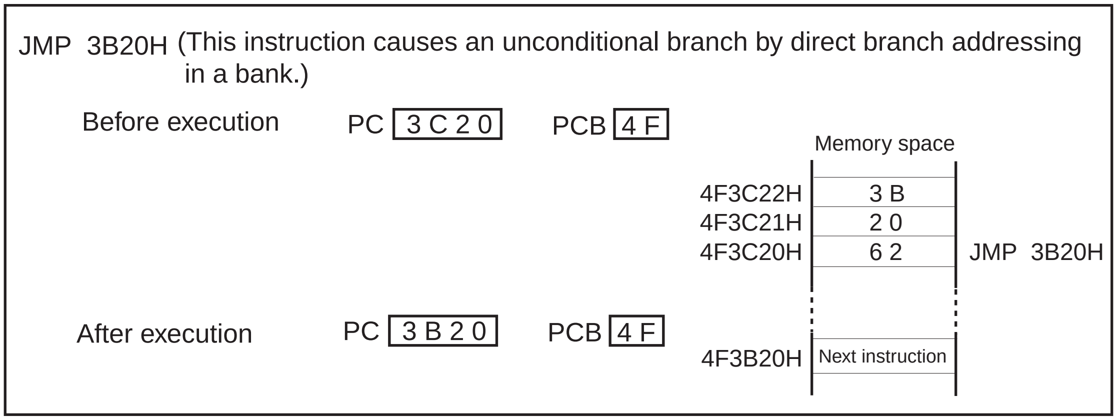
Figure B.3-2 Example of Register Direct Addressing


MOV R0, A (This instruction transfers the eight low-order bits of A to the general-purpose register R0.)
Before execution
A
0 7 1 6 2 5 3 4
After execution
A
0 7 1 6 2 5 6 4
Memory space
R0
? ?
Memory space
R0
3 4

- Direct branch addressing (addr16)

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bits 23 to 16 of the address are specified by the program bank register (PCB).

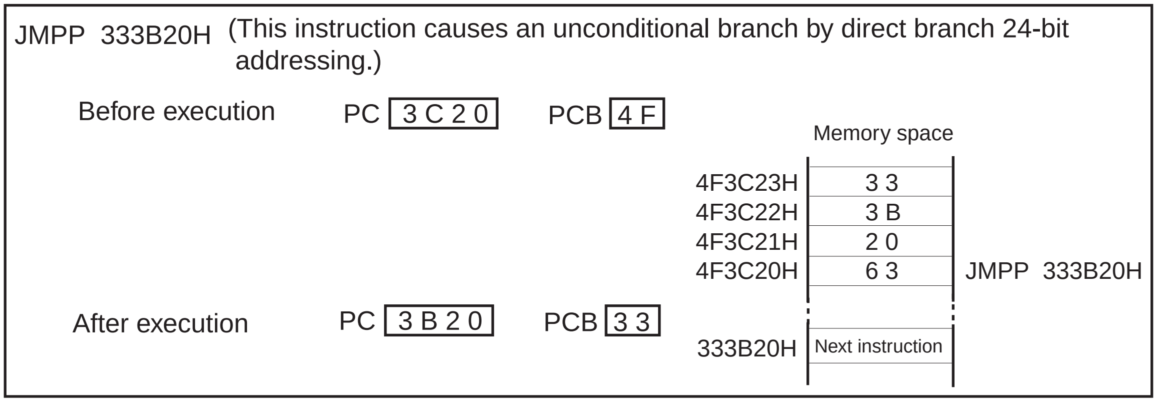
Figure B.3-3 Example of Direct Branch Addressing (addr16)



- Physical direct branch addressing (addr24)

Specify an offset explicitly for the branch destination address. The size of the offset is 24 bits. Physical direct branch addressing is used for unconditional branch, subroutine call, or software interrupt instruction.

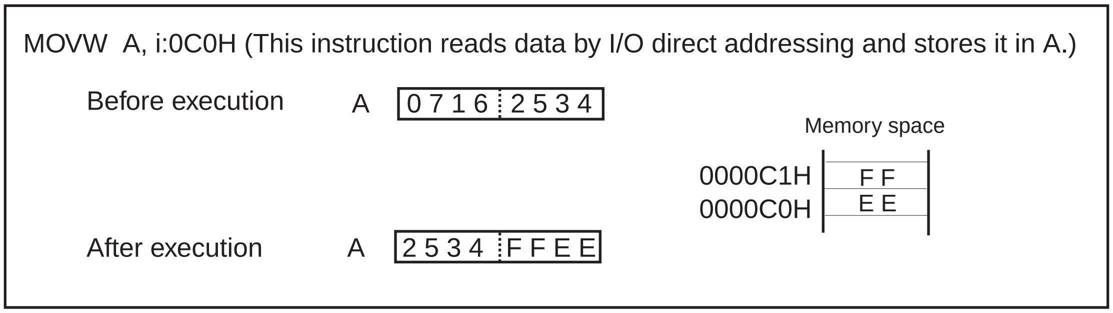
Figure B.3-4 Example of Direct Branch Addressing (addr24)



- I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000H to 0000FFH is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

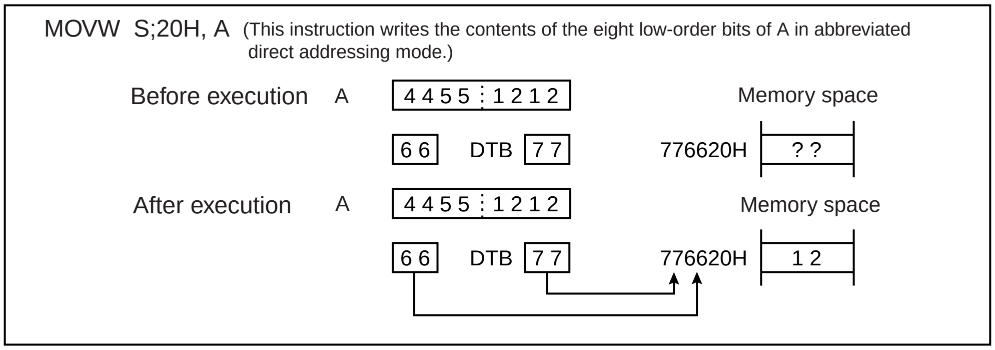
Figure B.3-5 Example of I/O Direct Addressing (io)



● Abbreviated direct addressing (dir)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB).

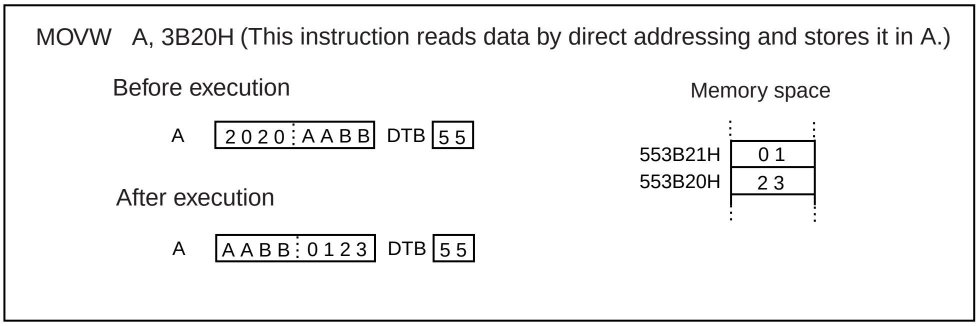
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)


MOVW S;20H, A (This instruction writes the contents of the eight low-order bits of A in abbreviated direct addressing mode.)
Before execution
A
4455 1212
Memory space
66
DTB
77
776620H
??
After execution
A
4455 1212
Memory space
66
DTB
77
776620H
12

● Direct addressing (addr16)

Specify the 16 low-order bits of a memory address explicitly in an operand. Address bits 16 to 23 are specified by the data bank register (DTB). A prefix instruction for access space addressing is invalid for this mode of addressing.

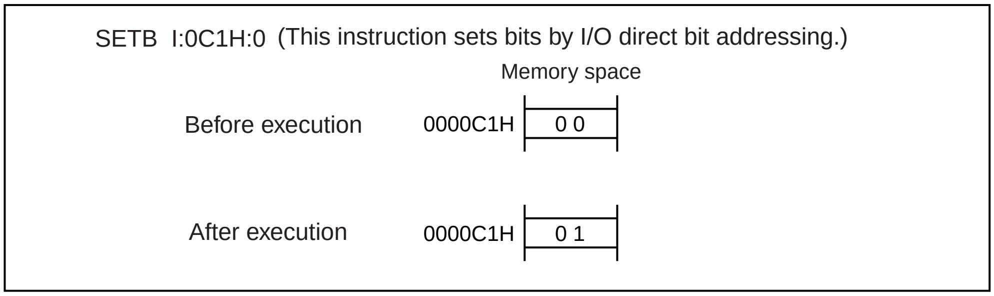
Figure B.3-7 Example of Direct Addressing (addr16)



● I/O direct bit addressing (io:bp)

Specify bits in physical addresses 000000_H to $0000FF_H$ explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

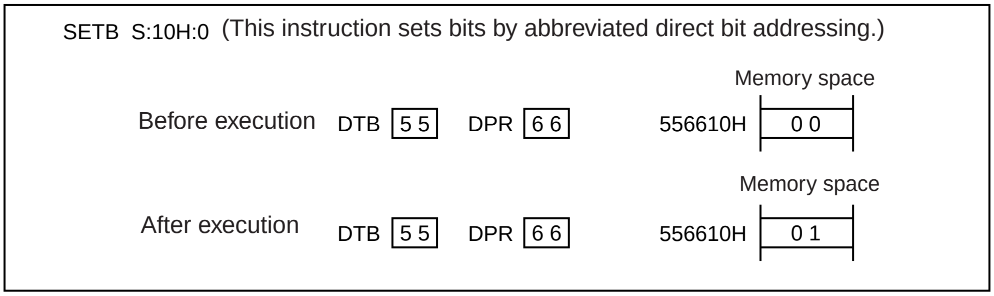
Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)


SETB I:0C1H:0 (This instruction sets bits by I/O direct bit addressing.)
Memory space
Before execution
0000C1H
0 0
After execution
0000C1H
0 1

● Abbreviated direct bit addressing (dir:bp)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

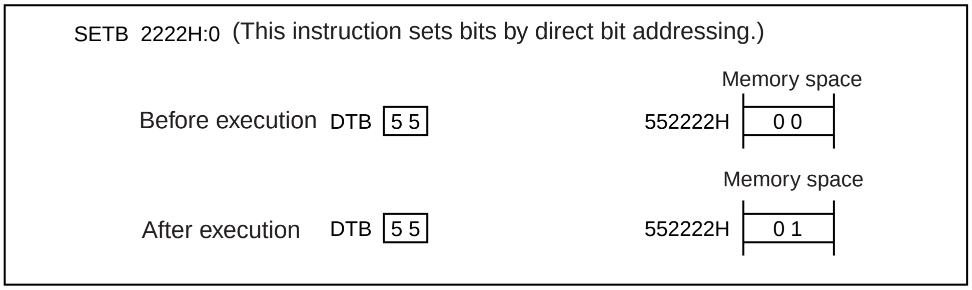
Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)



● Direct bit addressing (addr16:bp)

Specify arbitrary bits in 64K bytes explicitly. Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

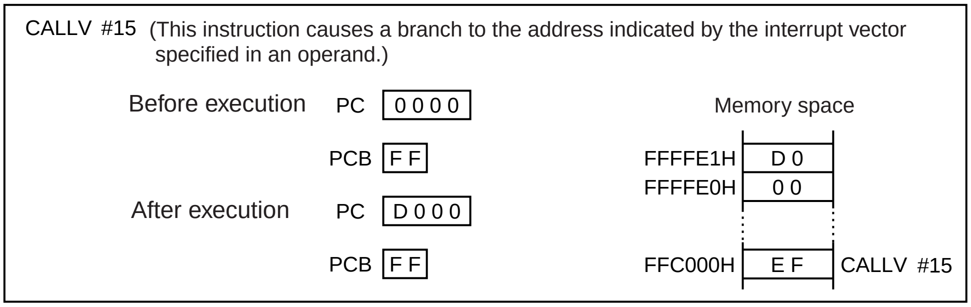
Figure B.3-10 Example of Direct Bit Addressing (addr16:bp)



● Vector Addressing (#vct)

Specify vector data in an operand to indicate the branch destination address. There are two sizes for vector numbers: 4 bits and 8 bits. Vector addressing is used for a subroutine call or software interrupt instruction.

Figure B.3-11 Example of Vector Addressing (#vct)



CALLV #15 (This instruction causes a branch to the address indicated by the interrupt vector specified in an operand.)

Table B.3-2 CALLV Vector List

Instruction	Vector address L	Vector address H
CALLV #0	XXFFFE$_H$	XXFFFF$_H$
CALLV #1	XXFFFC$_H$	XXFFFD$_H$
CALLV #2	XXFFFA$_H$	XXFFFB$_H$
CALLV #3	XXFFF8$_H$	XXFFF9$_H$
CALLV #4	XXFFF6$_H$	XXFFF7$_H$
CALLV #5	XXFFF4$_H$	XXFFF5$_H$
CALLV #6	XXFFF2$_H$	XXFFF3$_H$
CALLV #7	XXFFF0$_H$	XXFFF1$_H$
CALLV #8	XXFFEE$_H$	XXFFEF$_H$
CALLV #9	XXFFEC$_H$	XXFFED$_H$
CALLV #10	XXFFEA$_H$	XXFFEB$_H$
CALLV #11	XXFFE8$_H$	XXFFE9$_H$
CALLV #12	XXFFE6$_H$	XXFFE7$_H$
CALLV #13	XXFFE4$_H$	XXFFE5$_H$
CALLV #14	XXFFE2$_H$	XXFFE3$_H$
CALLV #15	XXFFE0$_H$	XXFFE1$_H$

Note: A PCB register value is set in XX.

Note:

When the program bank register (PCB) is FF$_H$, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3-2).

B.4 Indirect Addressing

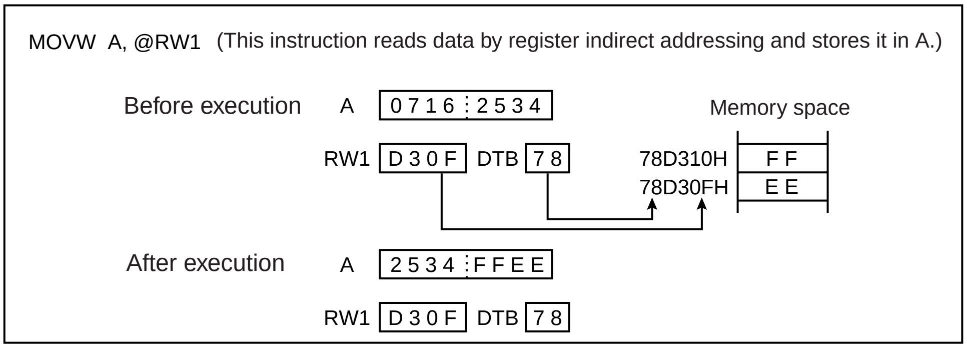
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ Indirect Addressing

● Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)

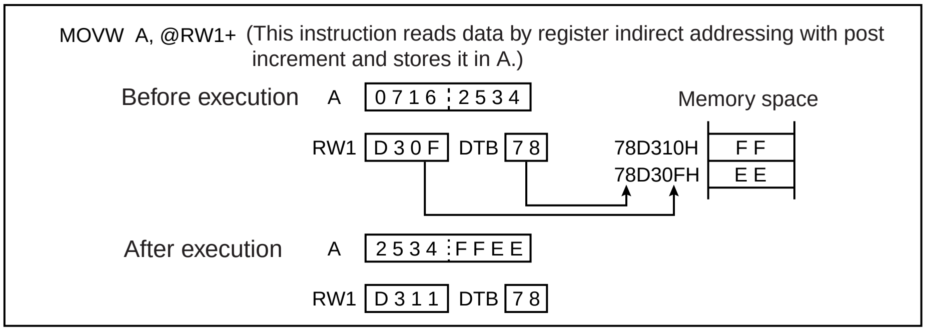


● Register indirect addressing with post increment (@RWj+ j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.

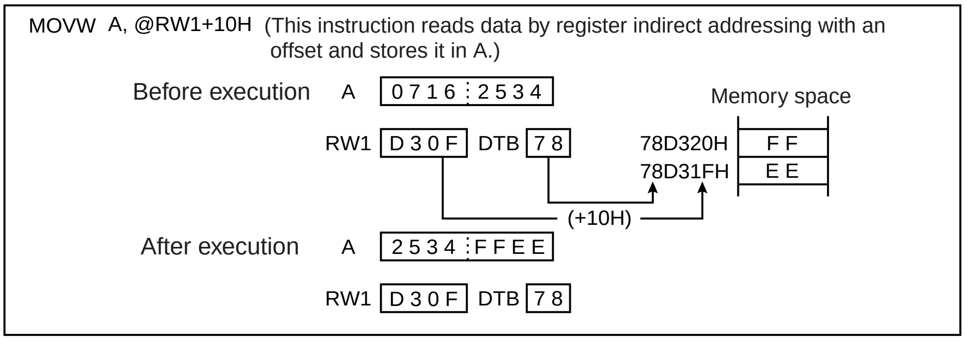
Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj+ j = 0 to 3)


MOVW A, @RW1+ (This instruction reads data by register indirect addressing with post increment and stores it in A.)
Before execution
A
0716 2534
Memory space
RW1
D30F
DTB
78
78D310H
FF
78D30FH
EE
After execution
A
2534 FFEE
RW1
D311
DTB
78

- Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

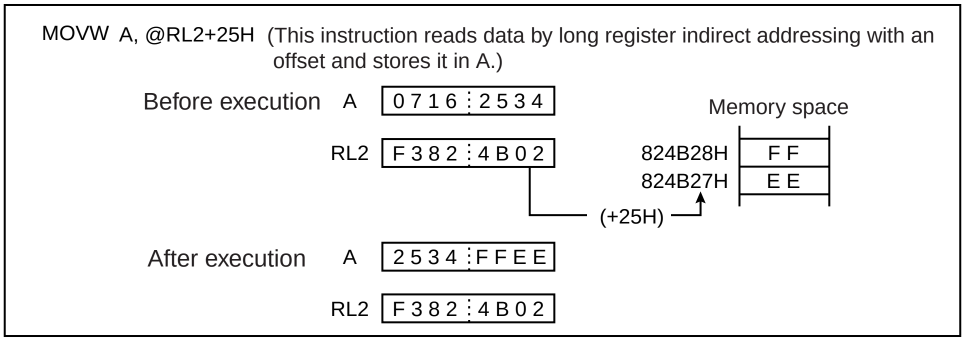
Figure B.4-3 Example of Register Indirect Addressing with Offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)


MOVW A, @RW1+10H (This instruction reads data by register indirect addressing with an offset and stores it in A.)
Before execution
A
0716 2534
Memory space
RW1
D30F
DTB
78
78D320H
FF
78D31FH
EE
(+10H)
After execution
A
2534 FFEE
RW1
D30F
DTB
78

- Long register indirect addressing with offset (@RLi + disp8 i = 0 to 3)

Memory is accessed using the address that is the 24 low-order bits obtained by adding an offset to the contents of general-purpose register RLi. The offset is 8-bits long and is added as a signed numeric value.

Figure B.4-4 Example of Long Register Indirect Addressing with Offset (@RLi + disp8 i = 0 to 3)

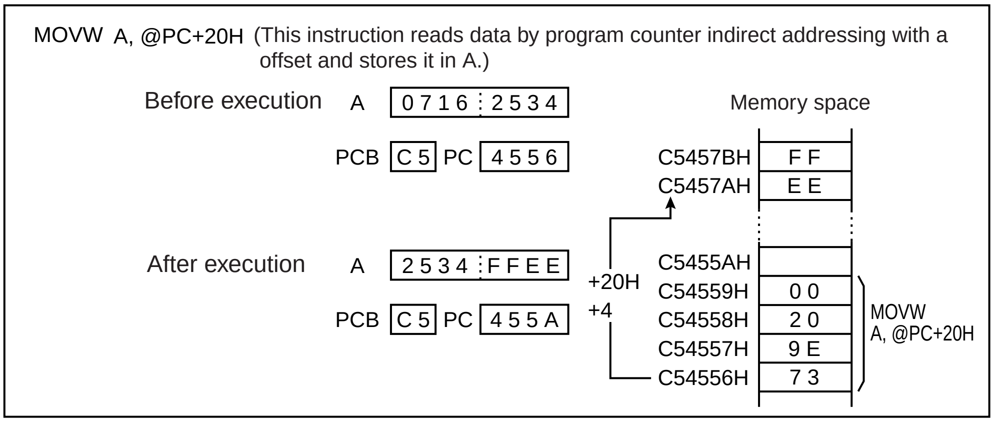

MOVW A, @RL2+25H (This instruction reads data by long register indirect addressing with an offset and stores it in A.)
Before execution
A
0716 2534
Memory space
RL2
F382 4B02
824B28H
FF
824B27H
EE
(+25H)
After execution
A
2534 FFEE
RL2
F382 4B02

● Program counter indirect addressing with offset (@PC + disp16)

Memory is accessed using the address indicated by (instruction address + 4 + disp16). The offset is one word long. Address bits 16 to 23 are specified by the program bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address + disp16):

- DBNZ eam, rel
- DWBNZ eam, rel
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel
- MOV eam, #imm8
- MOVW eam, #imm16

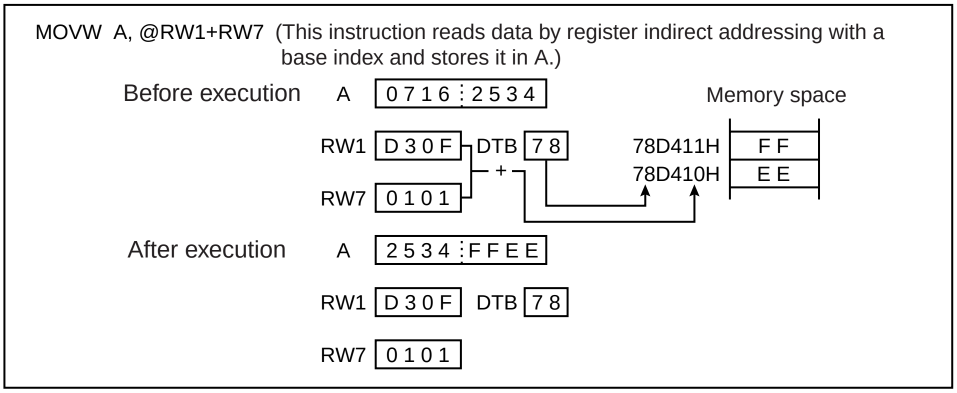
Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16)



● Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)

Memory is accessed using the address determined by adding RW0 or RW1 to the contents of general-purpose register RW7. Address bits 16 to 23 are indicated by the data bank register (DTB).

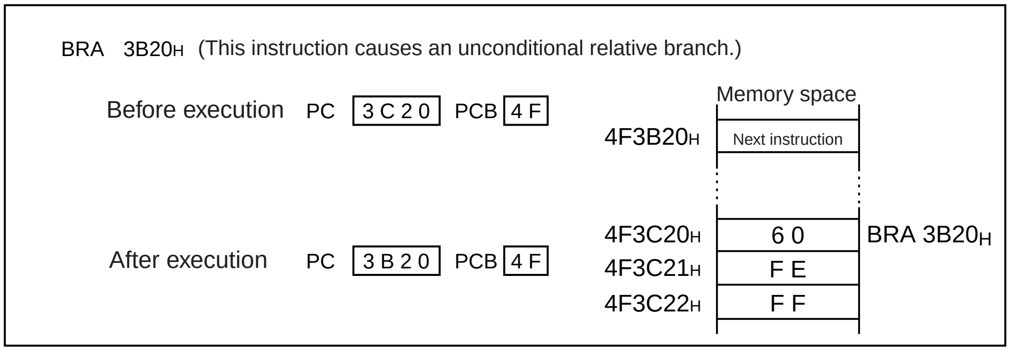
Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)



● Program counter relative branch addressing (rel)

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-Kbyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program bank register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



BRA 3B20H (This instruction causes an unconditional relative branch.)

● Register list (rlst)

Specify a register to be pushed onto or popped from a stack.

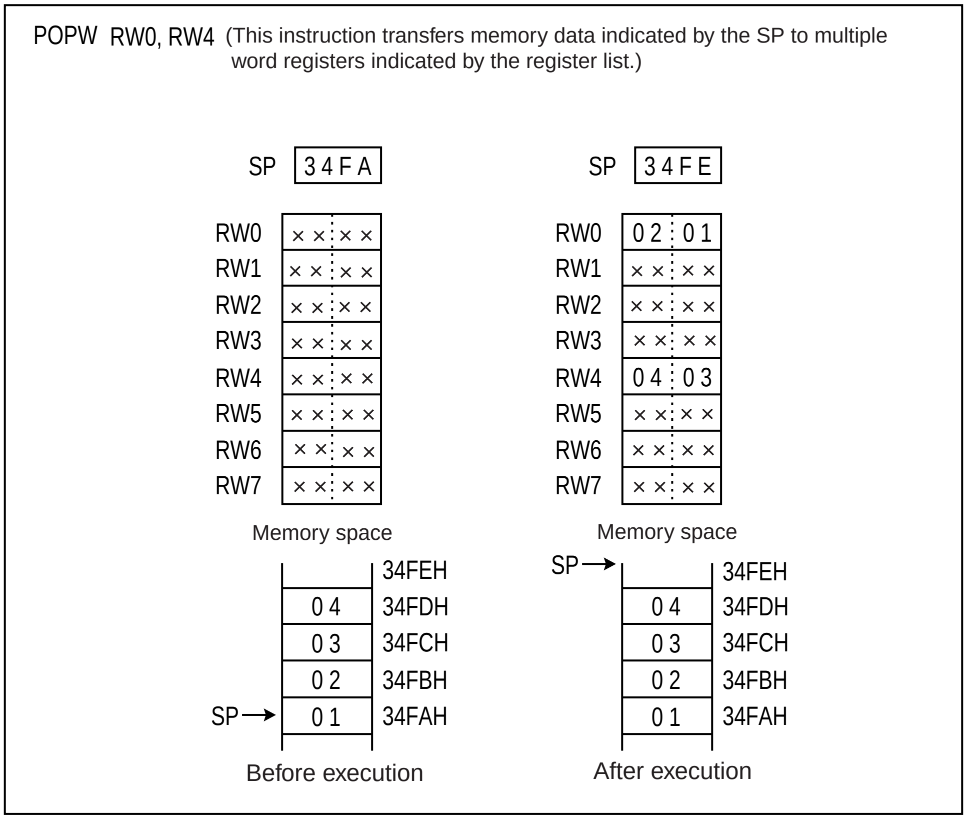
Figure B.4-8 Configuration of the Register List

MSB							LSB
RW7	RW6	RW5	RW4	RW3	RW2	RW1	RW0

A register is selected when the corresponding bit is 1 and deselected when the bit is 0.

MSB : Highest bit
LSB : Lowest bit

Figure B.4-9 Example of Register List (rlst)

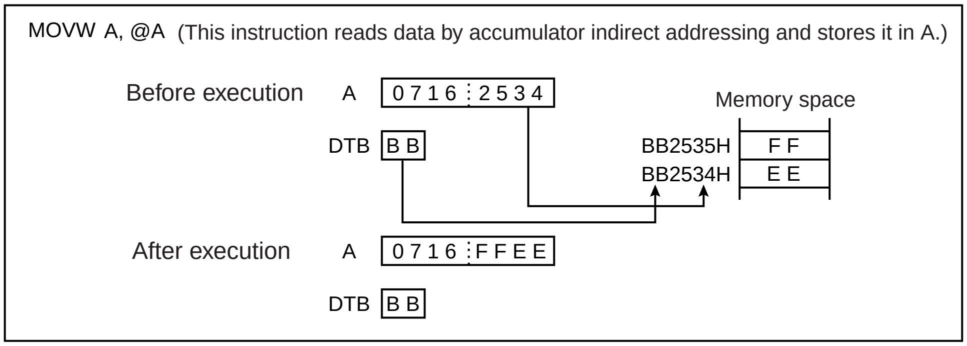


POPW RW0, RW4 (This instruction transfers memory data indicated by the SP to multiple word registers indicated by the register list.)

● Accumulator indirect addressing (@A)

Memory is accessed using the address indicated by the contents of the low-order bytes (16 bits) of the accumulator (AL). Address bits 16 to 23 are specified by a mnemonic in the data bank register (DTB).

Figure B.4-10 Example of Accumulator Indirect Addressing (@A)

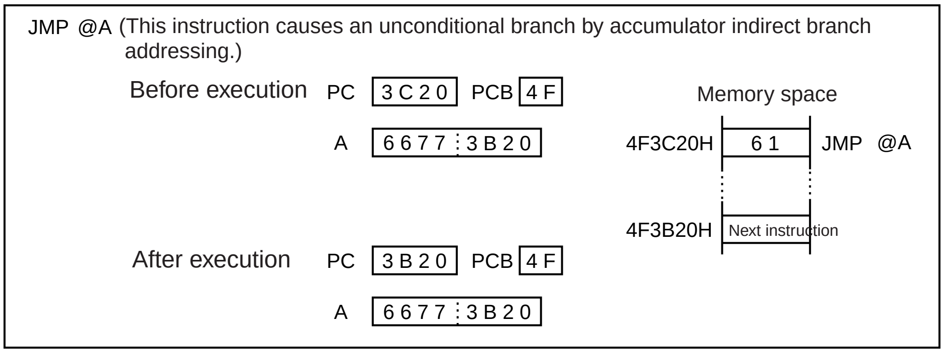


MOVW A, @A (This instruction reads data by accumulator indirect addressing and stores it in A.)

- Accumulator indirect branch addressing (@A)

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

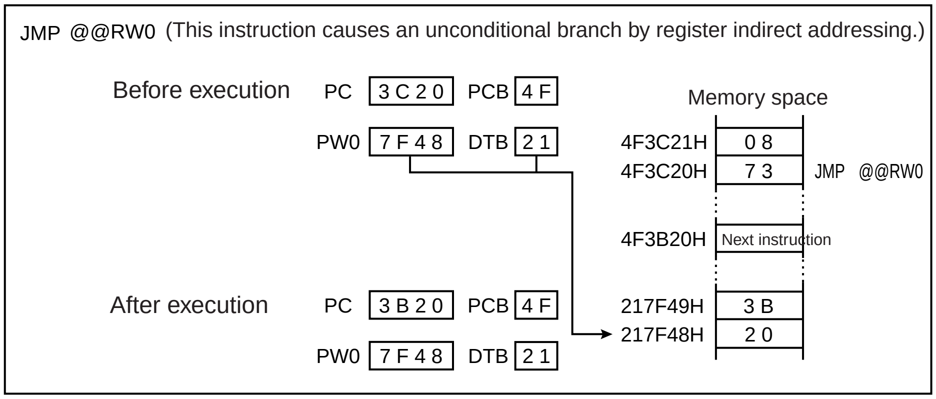
Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)



- Indirect specification branch addressing (@ear)

The address of the branch destination is the word data at the address indicated by ear.

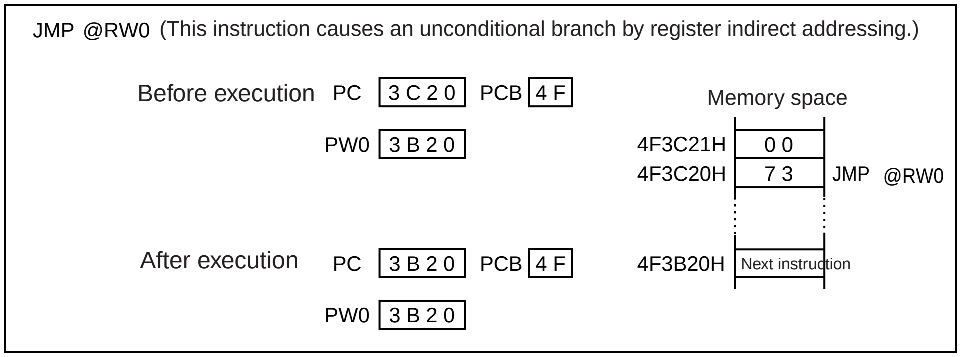
Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)



● Indirect specification branch addressing (@eam)

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



B.5 Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.

■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

■ Calculating the execution cycle count

Table B.5-1 lists execution cycle counts and Table B.5-2 and Table B.5-3 summarize correction value data.

Table B.5-1 Execution Cycle Counts in Each Addressing Mode

Code	Operand	(a) *	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00 | 07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08 | 0B	@RWj	2	1
0C | 0F	@RWj+	4	2
10 | 17	@RWi+disp8	2	1
18 | 1B	@RWi+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

*: (a) is used for ∼ (cycle count) and B (correction value) in "B.8 F^2MC-16LX Instruction List".

Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles

Operand	(b) byte [*1]		(c) word [*1]		(d) long [*1]	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus [*2] 16-bit even address	+1	1	+1	1	+2	2
External data bus [*2] 16-bit odd address	+1	1	+4	2	+8	4
External data bus [*2] 8-bits	+1	1	+4	2	+8	4

*1: (b), (c), and (d) are used for ∼ (cycle count) and B (correction value) in "B.8 F^{2}MC-16LX Instruction List".

*2: When an external data bus is used, the number of cycles during which an instruction is made to wait by ready - signal input or automatic ready must also be added.

Note:

When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	-	+2
External data bus 16-bits	-	+3
External data bus 8-bits	+3	-

Note:

- When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.
- Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.

B.6 Effective address field

Table B.6-1 shows the effective address field.

■ Effective Address Field

Table B.6-1 Effective Address Field

Code	Representation			Address format	Byte count of extended address part *
00	R0	RW0	RL0	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	-
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	0
09	@RW1				
0A	@RW2				
0B	@RW3				
0C	@RW0+			Register indirect with post increment	0
0D	@RW1+				
0E	@RW2+				
0F	@RW3+				
10	@RW0+disp8			Register indirect with 8-bit displacement	1
11	@RW1+disp8				
12	@RW2+disp8				
13	@RW3+disp8				
14	@RW4+disp8				
15	@RW5+disp8				
16	@RW6+disp8				
17	@RW7+disp8				
18	@RW0+disp16			Register indirect with 16-bit displacement	2
19	@RW1+disp16				
1A	@RW2+disp16				
1B	@RW3+disp16				
1C	@RW0+RW7			Register indirect with index	0
1D	@RW1+RW7			Register indirect with index	0
1E	@PC+disp16			PC indirect with 16-bit displacement	2
1F	addr16			Direct address	2

*: Each byte count of the extended address part applies to + in the # (byte count) column in "B.8 F^2MC-16LX Instruction List". For the meaning of "#", see "B.7 How to Read the Instruction List".

B.7 How to Read the Instruction List

Table B.7-1 describes the items used in the F^2MC-16LX Instruction List, and Table B.7-2 describes the symbols used in the same list.

■ Description of Instruction Presentation Items and Symbols

Table B.7-1 Description of Items in the Instruction List (1/2)

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number of following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bits 15 to 08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00_H or FF_H to AH after AL sign extension.
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change Z: Set upon instruction execution. X: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	

Table B.7-1 Description of Items in the Instruction List (2/2)

Item	Description
RMW	Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory). *: Read Modify Write instruction -: Not Read Modify Write instruction Note: Cannot be used for an address that has different meanings between read and write operations.

Table B.7-2 Explanation on Symbols in the Instruction List (1/2)

Symbol	Explanation
A	The bit length used varies depending on the 32-bit accumulator instruction. Byte: Low-order 8 bits of byte AL Word: 16 bits of word AL Long word: 32 bits of AL and AH
AH AL	16 high-order bits of A 16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	Program bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Abbreviated direct addressing
addr16 addr24 ad24 0-15 ad24 16-23	Direct addressing Physical direct addressing Bits 0 to 15 of addr24 Bits 16 to 23 of addr24
io	I/O area (000000$_H$ to 0000FF$_H$)

Table B.7-2 Explanation on Symbols in the Instruction List (2/2)

Symbol	Explanation
#imm4 #imm8 #imm16 #imm32 ext (imm8)	4-bit immediate data 8-bit immediate data 16-bit immediate data 32-bit immediate data 16-bit data obtained by sign extension of 8-bit immediate data
disp8 disp16	8-bit displacement 16-bit displacement
bp	Bit offset
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( ) b	Bit address
rel ear eam	PC relative branch Effective addressing (code 00 to 07) Effective addressing (code 08 to 1F)
rlst	Register list

B.8 F^2MC-16LX Instruction List

Table B.8-1 to Table B.8-18 list the instructions used by the F^2MC-16LX.

■ F^2MC-16LX Instruction List

Table B.8-1 41 Transfer Instructions (byte)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A,dir	2	3	0	(b)	byte (A) <-- (dir)	Z	*	-	-	*	*	-	-	-	-
MOV A,addr16	3	4	0	(b)	byte (A) <-- (addr16)	Z	*	-	-	*	*	-	-	-	-
MOV A,Ri	1	2	1	0	byte (A) <-- (Ri)	Z	*	-	-	*	*	-	-	-	-
MOV A,ear	2	2	1	0	byte (A) <-- (ear)	Z	*	-	-	*	*	-	-	-	-
MOV A,eam	2+	3 + (a)	0	(b)	byte (A) <-- (eam)	Z	*	-	-	*	*	-	-	-	-
MOV A,io	2	3	0	(b)	byte (A) <-- (io)	Z	*	-	-	*	*	-	-	-	-
MOV A,#imm8	2	2	0	0	byte (A) <-- imm8	Z	*	-	-	*	*	-	-	-	-
MOV A,@A	2	3	0	(b)	byte (A) <-- ((A))	Z	-	-	-	*	*	-	-	-	-
MOV A,@RLi+disp8	3	10	2	(b)	byte (A) <-- ((RLi)+disp8)	Z	*	-	-	*	*	-	-	-	-
MOVN A,#imm4	1	1	0	0	byte (A) <-- imm4	Z	*	-	-	R	*	-	-	-	-
MOVX A,dir	2	3	0	(b)	byte (A) <-- (dir)	X	*	-	-	-	*	*	-	-	-
MOVX A,addr16	3	4	0	(b)	byte (A) <-- (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX A,Ri	2	2	1	0	byte (A) <-- (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX A,ear	2	2	1	0	byte (A) <-- (ear)	X	*	-	-	-	*	*	-	-	-
MOVX A,eam	2+	3 + (a)	0	(b)	byte (A) <-- (eam)	X	*	-	-	-	*	*	-	-	-
MOVX A,io	2	3	0	(b)	byte (A) <-- (io)	X	*	-	-	-	*	*	-	-	-
MOVX A,#imm8	2	2	0	0	byte (A) <-- imm8	X	*	-	-	-	*	*	-	-	-
MOVX A,@A	2	3	0	(b)	byte (A) <-- ((A))	X	-	-	-	-	*	*	-	-	-
MOVX A,@RWi+disp8	2	5	1	(b)	byte (A) <-- ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@RLi+disp8	3	10	2	(b)	byte (A) <-- ((RLi)+disp8	X	*	-	-	-	*	*	-	-	-
MOV dir,A	2	3	0	(b)	byte (dir) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV addr16,A	3	4	0	(b)	byte (addr16) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,A	1	2	1	0	byte (Ri) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV ear,A	2	2	1	0	byte (ear) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV eam,A	2+	3 + (a)	0	(b)	byte (eam) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV io,A	2	3	0	(b)	byte (io) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV @RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,ear	2	3	2	0	byte (Ri) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOV Ri,eam	2+	4 + (a)	1	(b)	byte (Ri) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOV ear,Ri	2	4	2	0	byte (ear) <-- (Ri)	-	-	-	-	-	*	*	-	-	-
MOV eam,Ri	2+	5 + (a)	1	(b)	byte (eam) <-- (Ri)	-	-	-	-	-	*	*	-	-	-
MOV Ri,#imm8	2	2	1	0	byte (Ri) <-- imm8	-	-	-	-	-	*	*	-	-	-
MOV io,#imm8	3	5	0	(b)	byte (io) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV dir,#imm8	3	5	0	(b)	byte (dir) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV ear,#imm8	3	2	1	0	byte (ear) <-- imm8	-	-	-	-	-	*	*	-	-	-
MOV eam,#imm8	3+	4 + (a)	0	(b)	byte (eam) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV @AL,AH / MOV @A,T	2	3	0	(b)	byte ((A)) <-- (AH)	-	-	-	-	-	*	*	-	-	-
XCH A,ear	2	4	2	0	byte (A) <--> (ear)	Z	-	-	-	-	-	-	-	-	-
XCH A,eam	2+	5 + (a)	0	2 x (b)	byte (A) <--> (eam)	Z	-	-	-	-	-	-	-	-	-
XCH Ri,ear	2	7	4	0	byte (Ri) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCH Ri,eam	2+	9 + (a)	2	2 x (b)	byte (Ri) <--> (eam)	-	-	-	-	-	-	-	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-2 38 Transfer Instructions (byte)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,dir	2	3	0	(c)	word (A) <-- (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	4	0	(c)	word (A) <-- (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	3	1	0	0	word (A) <-- (SP)	-	*	-	-	-	*	*	-	-	-
MOVW A,RWi	1	2	1	0	word (A) <-- (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW A,ear	2	2	1	0	word (A) <-- (ear)	-	*	-	-	-	*	*	-	-	-
MOVW A,eam	2+	3 + (a)	0	(c)	word (A) <-- (eam)	-	*	-	-	-	*	*	-	-	-
MOVW A,io	2	3	0	(c)	word (A) <-- (io)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	3	0	(c)	word (A) <-- ((A))	-	-	-	-	-	*	*	-	-	-
MOVW A,#imm16	3	2	2	0	word (A) <-- imm16	-	*	-	-	-	*	*	-	-	-
MOVW A,@RWi+disp8	2	5	1	(c)	word (A) <-- ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@RLi+disp8	3	10	2	(c)	word (A) <-- ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW dir,A	2	3	0	(c)	word (dir) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW addr16,A	3	4	0	(c)	word (addr16) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW SP,A	1	1	0	0	word (SP) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,A	1	2	1	0	word (RWi) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW ear,A	2	2	1	0	word (ear) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW eam,A	2+	3 + (a)	0	(c)	word (eam) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW io,A	2	3	0	(c)	word (io) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RWi+disp8,A	2	5	1	(c)	word ((RWi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RLi+disp8,A	3	10	2	(c)	word ((RLi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,ear	2	3	2	0	word (RWi) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOVW	2+	4 + (a)	1	(c)	word (RWi) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVW ear,Rwi	2	4	2	0	word (ear) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW eam,Rwi	2+	5 + (a)	1	(c)	word (eam) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,#imm16	3	2	1	0	word (RWi) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW io,#imm16	4	5	0	(c)	word (io) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOVW ear,#imm16	4	2	1	0	word (ear) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW eam,#imm16	4+	4 + (a)	0	(c)	word (eam) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOVW @AL,AH / MOVW @A,T	2	3	0	(c)	word ((A)) <-- (AH)	-	-	-	-	-	*	*	-	-	-
XCHW A,ear	2	4	2	0	word (A) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW A,eam	2+	5 + (a)	0	2 x (c)	word (A) <-- >(eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi, ear	2	7	4	0	word (RWi) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi, eam	2+	9 + (a)	2	2 x (c)	word (RWi) <--> (eam)	-	-	-	-	-	-	-	-	-	-
MOVL A,ear	2	4	2	0	long (A) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOVL A,eam	2+	5 + (a)	0	(d)	long (A) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVL A,#imm32	5	3	0	0	long (A) <-- imm32	-	-	-	-	-	*	*	-	-	-
MOVL ear,A	2	4	2	0	long (ear1) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVL eam,A	2+	5 + (a)	0	(d)	long(eam1) <-- (A)	-	-	-	-	-	*	*	-	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-3 42 Addition/subtraction Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD	A,#imm8	2	2	0	0	byte (A) <-- (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD	A,dir	2	5	0	(b)	byte (A) <-- (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD	A,ear	2	3	1	0	byte (A) <-- (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD	A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD	ear,A	2	3	2	0	byte (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD	eam,A	2+	5 + (a)	0	2 x (b)	byte (eam) <-- (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC	A	1	2	0	0	byte (A) <-- (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC	A,ear	2	3	1	0	byte (A) <-- (A) + (ear)+ (C)	Z	-	-	-	-	*	*	*	*	-
ADDC	A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) + (eam)+ (C)	Z	-	-	-	-	*	*	*	*	-
ADDDC	A	1	3	0	0	byte (A) <-- (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB	A,#imm8	2	2	0	0	byte (A) <-- (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB	A,dir	2	5	0	(b)	byte (A) <-- (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB	A,ear	2	3	1	0	byte (A) <-- (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB	A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB	ear,A	2	3	2	0	byte (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB	eam,A	2+	5 + (a)	0	2 x (b)	byte (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC	A	1	2	0	0	byte (A) <-- (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC	A,ear	2	3	1	0	byte (A) <-- (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC	A,eam	2+	4 + (a)	0	(b)	byte (A) <-- (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC	A	1	3	0	0	byte (A) <-- (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW	A	1	2	0	0	word (A) <-- (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW	A,ear	2	3	1	0	word (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW	A,#imm16	3	2	0	0	word (A) <-- (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW	ear,A	2	3	2	0	word (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW	eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW	A,ear	2	3	1	0	word (A) <-- (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW	A	1	2	0	0	word (A) <-- (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW	A,ear	2	3	1	0	word (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW	A,#imm16	3	2	0	0	word (A) <-- (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW	ear,A	2	3	2	0	word (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW	eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW	A,ear	2	3	1	0	word (A) <-- (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL	A,ear	2	6	2	0	long (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL	A,#imm32	5	4	0	0	long (A) <-- (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL	A,ear	2	6	2	0	long (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL	A,#imm32	5	4	0	0	long (A) <-- (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-4 12 Increment/decrement Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2 x (d)	long (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2 x (d)	long (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-5 11 Compare Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-6 11 Unsigned Multiplication/division Instructions (word, long word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
DIVU	A	1	*1	0	0	word (AH) / byte (AL) quotient --> byte (AL) remainder --> byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU	A,ear	2	*2	1	0	word (A) / byte (ear) quotient --> byte (A) remainder --> byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU	A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient --> byte (A) remainder --> byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,ear	2	*4	1	0	long (A) / word (ear) quotient --> word (A) remainder --> word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,eam	2+	*5	0	*7	long (A) / word (eam) quotient --> word (A) remainder --> word (eam)	-	-	-	-	-	-	-	*	*	-
MULU	A	1	*8	0	0	byte (AH) * byte (AL) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,ear	2	*9	1	0	byte (A) * byte (ear) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A	1	*11	0	0	word (AH) * word (AL) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,ear	2	*12	1	0	word (A) * word (ear) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,eam	2+	*13	0	(c)	word (A) * word (eam) --> Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0 7: Overflow 15: Normal
*2: 4: Division by 0 8: Overflow 16: Normal
*3: 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal
*4: 4: Division by 0 7: Overflow 22: Normal
*5: 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal
*6: (b): Division by 0 or overflow 2 x (b): Normal
*7: (c): Division by 0 or overflow 2 x (c): Normal
*8: 3: Byte (AH) is 0. 7: Byte (AH) is not 0.
*9: 4: Byte (ear) is 0. 8: Byte (ear) is not 0.
*10: 5+(a): Byte (eam) is 0, 9+(a): Byte (eam) is not 0.
*11: 3: Word (AH) is 0. 11: Word (AH) is not 0.
*12: 4: Word (ear) is 0. 12: Word (ear) is not 0.
*13: 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-7 11 Signed Multiplication/division Instructions (word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV	A	2	*1	0	0	word (AH) / byte (AL) quotient --> byte (AL) remainder --> byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV	A,ear	2	*2	1	0	word (A) / byte (ear) quotient --> byte (A) remainder --> byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV	A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient --> byte (A) remainder --> byte (eam)	Z	-	-	-	-	-	-	*	*	-
DIVW	A,ear	2	*4	1	0	long (A) / word (ear) quotient --> word (A) remainder --> word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW	A,eam	2+	*5	0	*7	long (A) / word (eam) quotient --> word (A) remainder --> word (eam)	-	-	-	-	-	-	-	*	*	-
MUL	A	2	*8	0	0	byte (AH) * byte (AL) --> word (A)	-	-	-	-	-	-	-	-	-	-
MUL	A,ear	2	*9	1	0	byte (A) * byte (ear) --> word (A)	-	-	-	-	-	-	-	-	-	-
MUL	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULW	A	2	*11	0	0	word (AH) * word (AL) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULW	A,ear	2	*12	1	0	word (A) * word (ear) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULW	A,eam	2+	*13	0	(c)	word (A) * word (eam) --> Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3: Division by 0, 8 or 18: Overflow, 18: Normal
*2: 4: Division by 0, 11 or 22: Overflow, 23: Normal
*3: 5+(a): Division by 0, 12+(a) or 23+(a): Overflow, 24+(a): Normal
*4: When dividend is positive; 4: Division by 0, 12 or 30: Overflow, 31: Normal
When dividend is negative; 4: Division by 0, 12 or 31: Overflow, 32: Normal
*5: When dividend is positive; 5+(a): Division by 0, 12+(a) or 31+(a): Overflow, 32+(a): Normal
When dividend is negative; 5+(a): Division by 0, 12+(a) or 32+(a): Overflow, 33+(a): Normal
*6: (b): Division by 0 or overflow, 2 x (b): Normal
*7: (c): Division by 0 or overflow, 2 x (c): Normal
*8: 3: Byte (AH) is 0, 12: result is positive, 13: result is negative
*9: 4: Byte (ear) is 0, 13: result is positive, 14: result is negative
*10: 5+(a): Byte (eam) is 0, 14+(a): result is positive, 15+(a): result is negative
*11: 3: Word (AH) is 0, 16: result is positive, 19: result is negative
*12: 4: Word (ear) is 0, 17: result is positive, 20: result is negative
*13: 5+(a): Word (eam) is 0, 18+(a): result is positive, 21+(a): result is negative

Notes:

- The execution cycle count found when an overflow occurs in a DIV or DIVW instruction may be a pre-operation count or a post-operation count depending on the detection timing.
- When an overflow occurs with DIV or DIVW instruction, the contents of the AL are destroyed.
- See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-8 39 Logic 1 Instructions (byte, word)

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
AND	A,#imm8	2	2	0	0	byte (A) <-- (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND	A,ear	2	3	1	0	byte (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND	ear,A	2	3	2	0	byte (ear) <-- (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND	eam,A	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR	A,#imm8	2	2	0	0	byte (A) <-- (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR	A,ear	2	3	1	0	byte (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR	ear,A	2	3	2	0	byte (ear) <-- (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR	eam,A	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR	A,#imm8	2	2	0	0	byte (A) <-- (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR	A,ear	2	3	1	0	byte (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR	ear,A	2	3	2	0	byte (ear) <-- (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR	eam,A	2+	5+(a)	0	2 x (b)	byte (eam) <-- (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT	A	1	2	0	0	byte (A) <-- not (A)	-	-	-	-	-	*	*	R	-	-
NOT	ear	2	3	2	0	byte (ear) <-- not (ear)	-	-	-	-	-	*	*	R	-	-
NOT	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW	A	1	2	0	0	word (A) <-- (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	A,#imm16	3	2	0	0	word (A) <-- (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW	A,ear	2	3	1	0	word (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW	ear,A	2	3	2	0	word (ear) <-- (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW	A	1	2	0	0	word (A) <-- (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW	A,#imm16	3	2	0	0	word (A) <-- (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW	A,ear	2	3	1	0	word (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW	ear,A	2	3	2	0	word (ear) <-- (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW	eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW	A	1	2	0	0	word (A) <-- (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW	A,#imm16	3	2	0	0	word (A) <-- (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW	A,ear	2	3	1	0	word (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW	ear,A	2	3	2	0	word (ear) <-- (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW	eam,A	2+	5+(a)	0	2 x (c)	word (eam) <-- (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW	A	1	2	0	0	word (A) <-- not (A)	-	-	-	-	-	*	*	R	-	-
NOTW	ear	2	3	2	0	word (ear) <-- not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW	eam	2+	5+(a)	0	2 x (c)	word (eam) <-- not (eam)	-	-	-	-	-	*	*	R	-	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-9 6 Logic 2 Instructions (long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL	A,ear	2	6	2	0	long (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL	A,ear	2	6	2	0	long (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL	A,ear	2	6	2	0	long (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-10 6 Sign Inversion Instructions (byte, word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG	A	1	2	0	0	byte (A) <-- 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG	ear	2	3	2	0	byte (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW	A	1	2	0	0	word (A) <-- 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW	ear	2	3	2	0	word (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW	eam	2+	5+(a)	0	2 x (c)	word (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-11 1 Normalization Instruction (long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML	A,R0	2	*1	1	0	long (A) <-- Shifts to the position where '1' is set for the first time. byte (RD) <-- Shift count at that time	-	-	-	-	-	-	*	-	-	-

*1: 4 when all accumulators have a value of 0; otherwise, 6+(R0)

Table B.8-12 18 Shift Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC	A	2	2	0	0	byte (A) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	A	2	2	0	0	byte (A) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	ear	2	3	2	0	byte (ear) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	*
ROLC	ear	2	3	2	0	byte (ear) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	eam	2+	5+(a)	0	2 x (b)	byte (eam) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	*
ASR	A,R0	2	*1	1	0	byte (A) <-- Arithmetic right shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
LSR	A,R0	2	*1	1	0	byte (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
LSL	A,R0	2	*1	1	0	byte (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRW	A	1	2	0	0	word (A) <-- Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW	A/SHRW A	1	2	0	0	word (A) <-- Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW	A/SHLW A	1	2	0	0	word (A) <-- Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW	A,R0	2	*1	1	0	word (A) <-- Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW	A,R0	2	*1	1	0	word (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW	A,R0	2	*1	1	0	word (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL	A,R0	2	*2	1	0	long (A) <-- Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL	A,R0	2	*2	1	0	long (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL	A,R0	2	*2	1	0	long (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

*1: 6 when R0 is 0; otherwise, 5 + (R0)
*2: 6 when R0 is 0; otherwise, 6 + (R0)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-13 31 Branch 1 Instructions

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
BZ/BEQ	rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/BNE	rel	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLO	rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC/BHS	rel	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN	rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP	rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV	rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV	rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT	rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT	rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT	rel	2	*1	0	0	Branch on (V) nor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE	rel	2	*1	0	0	Branch on (V) nor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE	rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT	rel	2	*1	0	0	Branch on ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS	rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI	rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA	rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP	@A	1	2	0	0	word (PC) <-- (A)	-	-	-	-	-	-	-	-	-	-
JMP	addr16	3	3	0	0	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
JMP	@ear	2	3	1	0	word (PC) <-- (ear)	-	-	-	-	-	-	-	-	-	-
JMP	@eam	2+	4+(a)	0	(c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
JMPP	@ear *3	2	5	2	0	word (PC) <-- (ear), (PCB) <-- (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP	@eam *3	2+	6+(a)	0	(d)	word (PC) <-- (eam), (PCB) <-- (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP	addr24	4	4	0	0	word (PC) <-- ad24 0-15, (PCB) <-- ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL	@ear *4	2	6	1	(c)	word (PC) <-- (ear)	-	-	-	-	-	-	-	-	-	-
CALL	addr16 *5	2+	7+(a)	0	2 x (c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
CALL	@eam *4	3	6	0	(c)	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
CALLV	#vct4 *5	1	7	0	2 x (c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP	@ear *6	2	10	2	2 x (c)	word (PC) <-- (ear)0-15, (PCB) <-- (ear)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	@eam *6	2+	11+(a)	0	*2	word (PC) <-- (eam)0-15, (PCB) <-- (eam)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	addr24 *7	4	10	0	2 x (c)	word (PC) <-- addr0-15, (PCB) <-- addr16-23	-	-	-	-	-	-	-	-	-	-

*1: 4 when a branch is made; otherwise, 3
*2: 3 x (c) + (b)
*3: Read (word) of branch destination address
*4: W: Save to stack (word)   R: Read (word) of branch destination address
*5: Save to stack (word)
*6: W: Save to stack (long word), R: Read (long word) of branch destination address
*7: Save to stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-14 19 Branch 2 Instructions

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
CBNE	A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE	ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE	eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE	eam,#imm16,rel*9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ	ear,rel	3	*5	2	0	Branch on byte (ear) = (ear) - 1, (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ	eam,rel	3+	*6	2	2 x (b)	Branch on byte (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ	ear,rel	3	*5	2	0	Branch on word (ear) = (ear) - 1, (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ	eam,rel	3+	*6	2	2 x (c)	Branch on word (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT	#vct8	2	20	0	8 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT	addr16	3	16	0	6 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP	addr24	4	17	0	6 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9		1	20	0	8 x (c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI		1	*8	0	*7	Return from interrupt	-	-	*	*	*	*	*	*	*	-
LINK	#imm8	2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
UNLINK		1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET	*10	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP	*11	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

*1: 5 when a branch is made; otherwise, 4
*2: 13 when a branch is made; otherwise, 12
*3: 7+(a) when a branch is made; otherwise, 6+(a)
*4: 8 when a branch is made; otherwise, 7
*5: 7 when a branch is made; otherwise, 6
*6: 8+(a) when a branch is made; otherwise, 7+(a)
*7: 3 x (b) + 2 x (c) when jumping to the next interruption request; 6 x (c) when returning from the current interruption
*8: 15 when jumping to the next interruption request; 17 when returning from the current interruption
*9: Do not use RWj+ addressing mode with a CBNE or CWBNE instruction.
*10: Return from stack (word)
*11: Return from stack (long word)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-15 28 Other Control Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW	A	1	4	0	(c)	word (SP) <-- (SP) - 2, ((SP)) <-- (A)	-	-	-	-	-	-	-	-	-	-
PUSHW	AH	1	4	0	(c)	word (SP) <-- (SP) - 2, ((SP)) <-- (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW	PS	1	4	0	(c)	word (SP) <-- (SP) - 2, ((SP)) <-- (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW	rlst	2	*3	*5	*4	(SP) <-- (SP) - 2n, ((SP)) <-- (rlst)	-	-	-	-	-	-	-	-	-	-
POPW	A	1	3	0	(c)	word (A) <-- ((SP)), (SP) <-- (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW	AH	1	3	0	(c)	word (AH) <-- ((SP)), (SP) <-- (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW	PS	1	4	0	(c)	word (PS) <-- ((SP)), (SP) <-- (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW	rlst	2	*2	*5	*4	(rlst) <-- ((SP)), (SP) <-- (SP)	-	-	-	-	-	-	-	-	-	-
JCTX	@A	1	14	0	6 x (c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND	CCR,#imm8	2	3	0	0	byte (CCR) <-- (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR	CCR,#imm8	2	3	0	0	byte (CCR) <-- (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV	RP,#imm8	2	2	0	0	byte (RP) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	ILM,#imm8	2	2	0	0	byte (ILM) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,ear	2	3	1	0	word (RWi) <-- ear	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,eam	2+	2+(a)	1	0	word (RWi) <-- eam	-	-	-	-	-	-	-	-	-	-
MOVEA	A,ear	2	1	0	0	word (A) <-- ear	-	*	-	-	-	-	-	-	-	-
MOVEA	A,eam	2+	1+(a)	0	0	word (A) <-- eam	-	*	-	-	-	-	-	-	-	-
ADDSP	#imm8	2	3	0	0	word (SP) <-- ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP	#imm16	3	3	0	0	word (SP) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOV	A,brg1	2	*1	0	0	byte (A) <-- (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV	brg2,A	2	1	0	0	byte (brg2) <-- (A)	-	-	-	-	-	*	*	-	-	-
NOP		1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB		1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB		1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB		1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB		1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC		1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR		1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

*1: PCB, ADB, SSB, USB, SPB: 1, DTB, DPR: 2
*2: 7 + 3 x (POP count) + 2 x (POP last register number), 7 when RLST = 0 (no transfer register)
*3: 29 + 3 x (PUSH count) - 3 x (PUSH last register number), 8 when RLST = 0 (no transfer register)
*4: (POP count) x (c) or (PUSH count) x (c)
*5: (POP count) or (PUSH count)

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-16 21 Bit Operand Instructions

Mnemonic		#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
MOVB	A,dir:bp	3	5	0	(b)	byte (A) <-- (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,addr16:bp	4	5	0	(b)	byte (A) <-- (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,io:bp	3	4	0	(b)	byte (A) <-- (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	dir:bp,A	3	7	0	2 x (b)	bit (dir:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB	addr16:bp,A	4	7	0	2 x (b)	bit (addr16:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB	io:bp,A	3	6	0	2 x (b)	bit (io:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
SETB	dir:bp	3	7	0	2 x (b)	bit (dir:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB	addr16:bp	4	7	0	2 x (b)	bit (addr16:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB	io:bp	3	7	0	2 x (b)	bit (io:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
CLRB	dir:bp	3	7	0	2 x (b)	bit (dir:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB	addr16:bp	4	7	0	2 x (b)	bit (addr16:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB	io:bp	3	7	0	2 x (b)	bit (io:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
BBC	dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC	addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC	io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS	dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS	addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS	io:bp,rel	4	*1	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS	addr16:bp,rel	5	*3	0	2 x (b)	Branch on (addr16:bp) b = 1, bit = 1	-	-	-	-	-	-	*	-	-	*
WBTS	io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC	io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1: 8 when a branch is made; otherwise, 7
*2: 7 when a branch is made; otherwise, 6
*3: 10 when the condition is met; otherwise, 9
*4: Undefined count
*5: Until the condition is met

Note:

See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

Table B.8-17 6 Accumulator Operation Instructions (byte, word)

Mnemonic	#	~	RG	B	Operation	L H	A H	I	S	T	N	Z	V	C	R M W
SWAP	1	3	0	0	byte (A)0-7 <--> (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW	1	2	0	0	word (AH) <--> (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extension	-	z	-	-	-	R	*	-	-	-

Table B.8-18 10 String Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*5	*3	byte transfer @AH+ <-- @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*5	*3	byte transfer @AH- <-- @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*5	*4	byte search @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*5	*4	byte search @AH- <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	*5	*3	byte fill @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*5	*6	word transfer @AH+ <-- @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*5	*6	word transfer @AH- <-- @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*5	*7	word search @AH+ - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*5	*7	word search @AH- - AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	*5	*6	word fill @AH+ <-- AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, 4 + 7 x (RW0) when the counter expires, or 7n + 5 when a match occurs
*2: 5 when RW0 is 0; otherwise, 4 + 8 x (RW0)
*3: (b) x (RW0) + (b) x (RW0) When the source and destination access different areas, calculate the (b) item individually.
*4: (b) x n
*5: 2 x (RW0)
*6: (c) x (RW0) + (c) x (RW0) When the source and destination access different areas, calculate the (c) item individually.
*7: (c) x n

Note:

m: RW0 value (counter value), n: Loop count

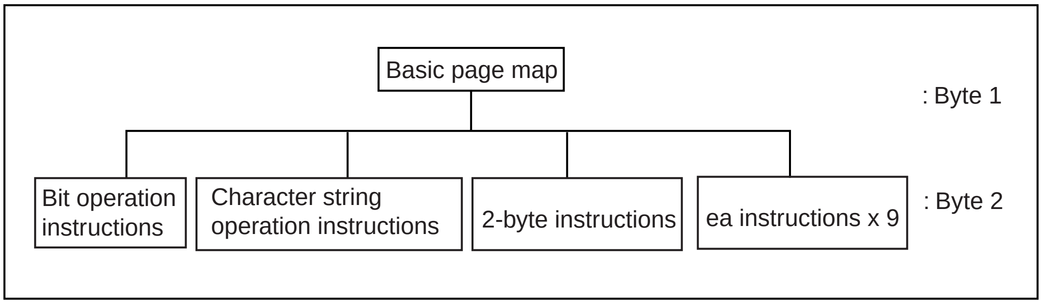
See Table B.5-1 and Table B.5-2 for information on (a) to (d) in the table.

B.9 Instruction Map

Each F^{2}MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Table B.9-2 to Table B.9-21 summarize the F^{2}MC-16LX instruction map.

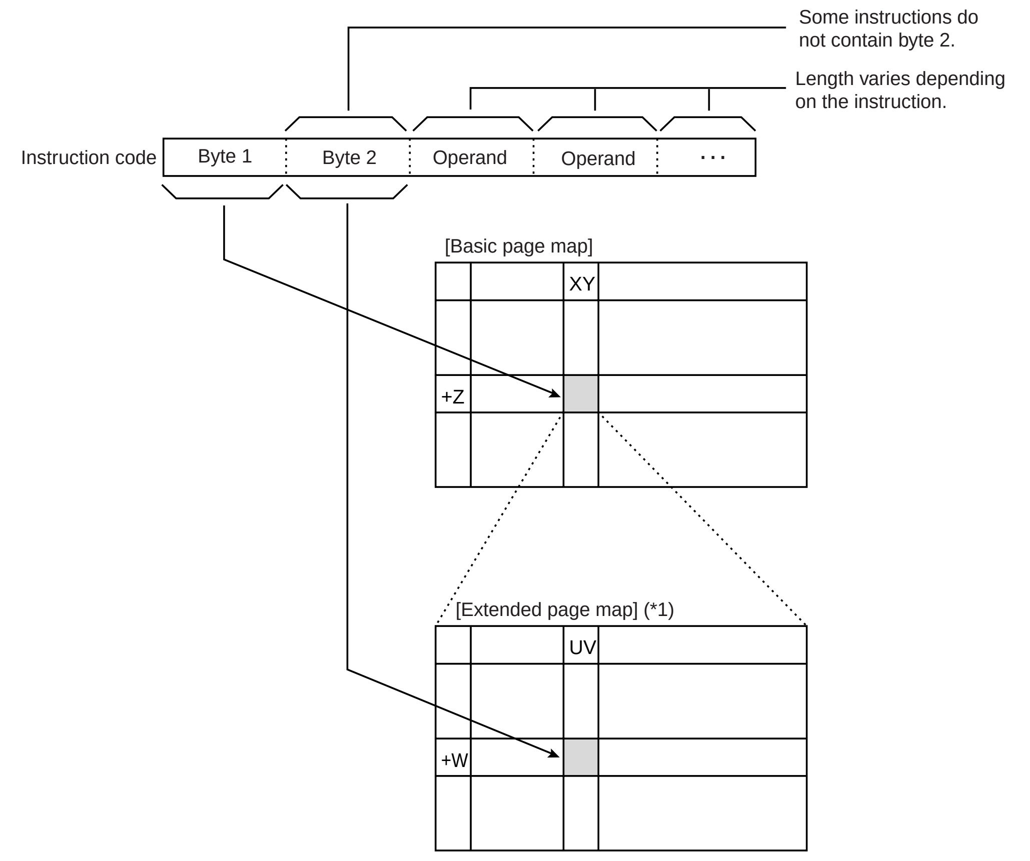
■ Structure of Instruction Map

Figure B.9-1 Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 shows the correspondence between an actual instruction code and instruction map.

Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map



*1 The extended page map is a generic name of maps for bit operation instructions, character string operation instructions, 2-byte instructions, and ea instructions. Actually, there are multiple extended page maps for each type of instructions.

An example of an instruction code is shown in Table B.9-1 .

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-
MOV A, ADB	60 +F=6F	00 +0=00
@RW2+d8, #8rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	NOP	CMR	ADD A,dir	ADD A,#8	MOV A,dir	MOV A,io	BRA rel	ea instruction 1	MOV A,Ri	MOV Ri,A	MOV Ri,#8	MOVX A,Ri	MOVX A, @RWi+d8	MOVN A,#4	CALLV #4	BZ /BEQ rel
+ 1	INT9	NCC	SUB A,dir	SUB A,#8	MOV dir,A	MOV io,A	JMP @A	ea instruction 2	↓	↓	↓	↓	↓	↓	↓	BNZ/BNE rel
+ 2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A,#8	MOV A,addr16	JMP addr16	ea instruction 3	↓	↓	↓	↓	↓	↓	↓	BC /BLO rel
+ 3	NEG A	JCTX @A	CMP A	CMP A,#8	MOVX A,#8	MOV addr16,A	JMPP addr24	ea instruction 4	↓	↓	↓	↓	↓	↓	↓	BNC /BHS rel
+ 4	PCB	EXT	AND CCR,#8	AND A,#8	MOV dir,#8	MOV io,#8	CALL addr16	ea instruction 5	↓	↓	↓	↓	↓	↓	↓	BN rel
+ 5	DTB	ZEXT	OR CCR,#8	OR A,#8	MOVX A,dir	MOVX A,io	CALLP addr24	ea instruction 6	↓	↓	↓	↓	↓	↓	↓	BP rel
+ 6	ADB	SWAP	DIVU A	XOR A,#8	MOVW A,SP	MOVW io,#16	RETP	ea instruction 7	↓	↓	↓	↓	↓	↓	↓	BV rel
+ 7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP,A	MOVX A,addr16	RET	ea instruction 8	↓	↓	↓	↓	↓	↓	↓	BNV rel
+ 8	LINK #imm8	ADDL A,#32	ADDW A	ADDW A,#16	MOVW A,dir	MOVW A,io	INT #vct8	ea instruction 9	MOVW A,RWi	MOVW RWi,A	MOVW RWi,#16	MOVW A, @RWi+d8	MOVW @RWi+d8,A	↓	↓	BT rel
+ 9	UNLINK	SUBL A,#32	SUBW A	SUBW A,#16	MOVW dir,A	MOVW io,A	INT addr16	MOVEA RWi,ea	↓	↓	↓	↓	↓	↓	↓	BNT rel
+ A	MOV RP,#8	MOV ILM,#8	CBNE A, #8,rel	CWBNE A, #16,rel	MOVW A,#16	MOVW A,addr16	INTP addr24	MOV **Ri,ea**	↓	↓	↓	↓	↓	↓	↓	BLT rel
+ B	NEGW	CMPL A,#32	CMPW A	CMPW A,#16	MOVL A,#32	MOVW addr16,A	RETI	MOVW RWi,ea	↓	↓	↓	↓	↓	↓	↓	BGE rel
+ C	LSLW A	EXTW	ANDW A	ANDW A,#16	PUSHW A	POPW A	Bit operation instruction	MOV ea,Ri	↓	↓	↓	↓	↓	↓	↓	BLE rel
+ D		ZEXTW	ORW A	ORW A,#16	PUSHW AH	POPW AH		MOVW ea,RWi	↓	↓	↓	↓	↓	↓	↓	BGT rel
+ E	ASRW A	SWAPW	XORW A	XORW A,#16	PUSHW PS	POPW PS	Character string operation instruction	XCH Ri,ea	↓	↓	↓	↓	↓	↓	↓	BLS rel
+ F	LSRW A	ADDSP #16	MULUW A	NOTW A	PUSHW rlst	POPW rlst	2-byte instruction	XCHW RWi,ea	↓	↓	↓	↓	↓	↓	↓	BHI rel

Table B.9-3 Bit Operation Instruction Map (first byte = 6C$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVB A,io:bp		MOVB io:bp,A		CLRB io:bp		SETB io:bp		BBC io:bp,rel		BBS io:bp,rel		WBTS io:bp		WBTC io:bp	
+ 1																
+ 2																
+ 3																
+ 4																
+ 5																
+ 6																
+ 7																
+ 8	MOVB A,dir:bp	MOVB A,addr16:bp	MOVB dir:bp,A	MOVB addr16:bp,A	CLRB dir:bp	CLRB addr16:bp	SETB dir:bp	SETB addr16:bp	BBC dir:bp,rel	BBC ad16:bp,rel	BBS dir:bp,rel	BBS ad16:bp,rel				SBBS addr16:bp
+ 9																
+ A																
+ B																
+ C																
+ D																
+ E																
+ F																

Table B.9-4 Character String Operation Instruction Map (first byte = 6E_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI PCB,PCB	MOVSD	MOVSWI	MOVSWD					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB		FILSWI PCB	
+1	PCB,DTB								DTB	DTB	DTB	DTB	DTB		DTB	
+2	PCB,ADB								ADB	ADB	ADB	ADB	ADB		ADB	
+3	PCB,SPB								SPB	SPB	SPB	SPB	SPB		SPB	
+4	DTB,PCB															
+5	DTB,DTB															
+6	DTB,ADB															
+7	DTB,SPB															
+8	ADB,PCB															
+9	ADB,DTB															
+A	ADB,ADB															
+B	ADB,SPB															
+C	SPB,PCB															
+D	SPB,DTB															
+E	SPB,ADB															
+F	SPB,SPB															

Table B.9-5 2-byte Instruction Map (first byte = 6F$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A,DTB	MOV DTB,A	MOVX A, @RL0+d8	MOV @RL0 +d8,A	MOV A, @RL0+d8											
+1	MOV A,ADB	MOV ADB,A														
+2	MOV A,SSB	MOV SSB,A	MOVX A, @RL1+d8	MOV @RL1 +d8,A	MOV A, @RL1+d8											
+3	MOV A,USB	MOV USB,A														
+4	MOV A,DPR	MOV DPR,A	MOVX A, @RL2+d8	MOV @RL2 +d8,A	MOV A, @RL2+d8											
+5	MOV A,@A	MOV @AL,AH														
+6	MOV A,PCB	MOVX A,@A	MOVX A, @RL3+d8	MOV @RL3 +d8,A	MOV A, @RL3+d8											
+7	ROLC A	RORC A														
+8				MOVW @RL0+d8,A	MOVW A, @RL0+d8		MUL A									
+9							MULW A									
+A				MOVW @RL1+d8,A	MOVW A, @RL1+d8		DIVU A									
+B																
+C	LSLW A,R0	LSLL A,R0	LSL A,R0	MOVW @RL2+d8,A	MOVW A, @RL2+d8											
+D	MOVW A,@A	MOVW @AL,AH	NRML A,R0													
+E	ASRW A,R0	ASRL A,R0	ASR A,R0	MOVW @RL3+d8,A	MOVW A, @RL3+d8											
+F	LSRW A,R0	LSRL A,R0	LSR A,R0													

Table B.9-6 ea Instruction 1 (first byte = 70_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBNE ↓	CWBNE ↓									CBNE ↓	CBNE ↓
+ 0	ADDL A,RL0	ADDL A, @RW0+d8	SUBL A,RL0	SUBL A, @RW0+d8	RW0, #16,rel	@RW0+d8, #16,rel	CMPL A,RL0	CMPL A, @RW0+d8	ANDL A,RL0	ANDL A, @RW0+d8	ORL A,RL0	ORL A, @RW0+d8	XORL A,RL0	XORL A, @RW0+d8	R0, #8,rel	@RW0+d8, #8,rel
+ 1	ADDL A,RL0	ADDL A, @RW1+d8	SUBL A,RL0	SUBL A, @RW1+d8	RW1, #16,rel	@RW1+d8, #16,rel	CMPL A,RL0	CMPL A, @RW1+d8	ANDL A,RL0	ANDL A, @RW1+d8	ORL A,RL0	ORL A, @RW1+d8	XORL A,RL0	XORL A, @RW1+d8	R1, #8,rel	@RW1+d8, #8,rel
+ 2	ADDL A,RL1	ADDL A, @RW2+d8	SUBL A,RL1	SUBL A, @RW2+d8	RW2, #16,rel	@RW2+d8, #16,rel	CMPL A,RL1	CMPL A, @RW2+d8	ANDL A,RL1	ANDL A, @RW2+d8	ORL A,RL1	ORL A, @RW2+d8	XORL A,RL1	XORL A, @RW2+d8	R2, #8,rel	@RW2+d8, #8,rel
+ 3	ADDL A,RL1	ADDL A, @RW3+d8	SUBL A,RL1	SUBL A, @RW3+d8	RW3, #16,rel	@RW3+d8, #16,rel	CMPL A,RL1	CMPL A, @RW3+d8	ANDL A,RL1	ANDL A, @RW3+d8	ORL A,RL1	ORL A, @RW3+d8	XORL A,RL1	XORL A, @RW3+d8	R3, #8,rel	@RW3+d8, #8,rel
+ 4	ADDL A,RL2	ADDL A, @RW4+d8	SUBL A,RL2	SUBL A, @RW4+d8	RW4, #16,rel	@RW4+d8, #16,rel	CMPL A,RL2	CMPL A, @RW4+d8	ANDL A,RL2	ANDL A, @RW4+d8	ORL A,RL2	ORL A, @RW4+d8	XORL A,RL2	XORL A, @RW4+d8	R4, #8,rel	@RW4+d8, #8,rel
+ 5	ADDL A,RL2	ADDL A, @RW5+d8	SUBL A,RL2	SUBL A, @RW5+d8	RW5, #16,rel	@RW5+d8, #16,rel	CMPL A,RL2	CMPL A, @RW5+d8	ANDL A,RL2	ANDL A, @RW5+d8	ORL A,RL2	ORL A, @RW5+d8	XORL A,RL2	XORL A, @RW5+d8	R5, #8,rel	@RW5+d8, #8,rel
+ 6	ADDL A,RL3	ADDL A, @RW6+d8	SUBL A,RL3	SUBL A, @RW6+d8	RW6, #16,rel	@RW6+d8, #16,rel	CMPL A,RL3	CMPL A, @RW6+d8	ANDL A,RL3	ANDL A, @RW6+d8	ORL A,RL3	ORL A, @RW6+d8	XORL A,RL3	XORL A, @RW6+d8	R6, #8,rel	@RW6+d8, #8,rel
+ 7	ADDL A,RL3	ADDL A, @RW7+d8	SUBL A,RL3	SUBL A, @RW7+d8	RW7, #16,rel	@RW7+d8, #16,rel	CMPL A,RL3	CMPL A, @RW7+d8	ANDL A,RL3	ANDL A, @RW7+d8	ORL A,RL3	ORL A, @RW7+d8	XORL A,RL3	XORL A, @RW7+d8	R7, #8,rel	@RW7+d8, #8,rel
+ 8	ADDL A,@RW0	ADDL A, @RW0+d16	SUBL A,@RW0	SUBL A, @RW0+d16	@RW0, #16,rel	@RW0+d16 , #16,rel	CMPL A,@RW0	CMPL A, @RW0+d16	ANDL A,@RW0	ANDL A, @RW0+d16	ORL A,@RW0	ORL A, @RW0+d16	XORL A,@RW0	XORL A, @RW0+d16	@RW0, #8,rel	@RW0+d16 ,#8,rel
+ 9	ADDL A,@RW1	ADDL A, @RW1+d16	SUBL A,@RW1	SUBL A, @RW1+d16	@RW1, #16,rel	@RW1+d16 , #16,rel	CMPL A,@RW1	CMPL A, @RW1+d16	ANDL A,@RW1	ANDL A, @RW1+d16	ORL A,@RW1	ORL A, @RW1+d16	XORL A,@RW1	XORL A, @RW1+d16	@RW1, #8,rel	@RW1+d16 ,#8,rel
+ A	ADDL A,@RW2	ADDL A, @RW2+d16	SUBL A,@RW2	SUBL A, @RW2+d16	@RW2, #16,rel	@RW2+d16 , #16,rel	CMPL A,@RW2	CMPL A, @RW2+d16	ANDL A,@RW2	ANDL A, @RW2+d16	ORL A,@RW2	ORL A, @RW2+d16	XORL A,@RW2	XORL A, @RW2+d16	@RW2, #8,rel	@RW2+d16 ,#8,rel
+ B	ADDL A,@RW3	ADDL A, @RW3+d16	SUBL A,@RW3	SUBL A, @RW3+d16	@RW3, #16,rel	@RW3+d16 , #16,rel	CMPL A,@RW3	CMPL A, @RW3+d16	ANDL A,@RW3	ANDL A, @RW3+d16	ORL A,@RW3	ORL A, @RW3+d16	XORL A,@RW3	XORL A, @RW3+d16	@RW3, #8,rel	@RW3+d16 ,#8,rel
+ C	ADDL A,@RW0+	ADDL A, @RW0+RW7	SUBL A,@RW0+	SUBL A, @RW0+RW7	Use prohibited	@RW0+RW7 , #16,rel	CMPL A,@RW0+	CMPL A, @RW0+RW7	ANDL A,@RW0+	ANDL A, @RW0+RW7	ORL A,@RW0+	ORL A, @RW0+RW7	XORL A,@RW0+	XORL A, @RW0+RW7	Use prohibited	@RW0+RW7 ,#8,rel
+ D	ADDL A,@RW1+	ADDL A, @RW1+RW7	SUBL A,@RW1+	SUBL A, @RW1+RW7	Use prohibited	@RW1+RW7 , #16,rel	CMPL A,@RW1+	CMPL A, @RW1+RW7	ANDL A,@RW1+	ANDL A, @RW1+RW7	ORL A,@RW1+	ORL A, @RW1+RW7	XORL A,@RW1+	XORL A, @RW1+RW7	Use prohibited	@RW1+RW7 ,#8,rel
+ E	ADDL A,@RW2+	ADDL A, @PC+d16	SUBL A,@RW2+	SUBL A, @PC+d16	Use prohibited	@PC+d16, #16,rel	CMPL A,@RW2+	CMPL A, @PC+d16	ANDL A,@RW2+	ANDL A, @PC+d16	ORL A,@RW2+	ORL A, @PC+d16	XORL A,@RW2+	XORL A, @PC+d16	Use prohibited	@PC+d16, #8,rel
+ F	ADDL A,@RW3+	ADDL A, addr16	SUBL A,@RW3+	SUBL A, addr16	Use prohibited	addr16, #16,rel	CMPL A,@RW3+	CMPL A, addr16	ANDL A,@RW3+	ANDL A, addr16	ORL A,@RW3+	ORL A, addr16	XORL A,@RW3+	XORL A, addr16	Use prohibited	addr16, #8,rel

Table B.9-7 ea Instruction 2 (first byte = 71_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMPP @RL0	JMPP @@RW0+d8	CALLP @RL0	CALLP @@RW0+d8	INCL RL0	INCL @RW0+d8	DECL RL0	DECL @RW0+d8	MOVL A,RL0	MOVL A, @RW0+d8	MOVL RL0,A	MOVL @R W0+d8,A	MOV R0,#8	MOV @R W0+d8,#8	MOVEA A,RW0	MOVEA A, @RW0+d8
+ 1	JMPP @RL0	JMPP @@RW1+d8	CALLP @RL0	CALLP @@RW1+d8	INCL RL0	INCL @RW1+d8	DECL RL0	DECL @RW1+d8	MOVL A,RL0	MOVL A, @RW1+d8	MOVL RL0,A	MOVL @R W1+d8,A	MOV R1,#8	MOV @R W1+d8,#8	MOVEA A,RW1	MOVEA A, @RW1+d8
+ 2	JMPP @RL1	JMPP @@RW2+d8	CALLP @RL1	CALLP @@RW2+d8	INCL RL1	INCL @RW2+d8	DECL RL1	DECL @RW2+d8	MOVL A,RL1	MOVL A, @RW2+d8	MOVL RL1,A	MOVL @R W2+d8,A	MOV R2,#8	MOV @R W2+d8,#8	MOVEA A,RW2	MOVEA A, @RW2+d8
+ 3	JMPP @RL1	JMPP @@RW3+d8	CALLP @RL1	CALLP @@RW3+d8	INCL RL1	INCL @RW3+d8	DECL RL1	DECL @RW3+d8	MOVL A,RL1	MOVL A, @RW3+d8	MOVL RL1,A	MOVL @R W3+d8,A	MOV R3,#8	MOV @R W3+d8,#8	MOVEA A,RW3	MOVEA A, @RW3+d8
+ 4	JMPP @RL2	JMPP @@RW4+d8	CALLP @RL2	CALLP @@RW4+d8	INCL RL2	INCL @RW4+d8	DECL RL2	DECL @RW4+d8	MOVL A,RL2	MOVL A, @RW4+d8	MOVL RL2,A	MOVL @R W4+d8,A	MOV R4,#8	MOV @R W4+d8,#8	MOVEA A,RW4	MOVEA A, @RW4+d8
+ 5	JMPP @RL2	JMPP @@RW5+d8	CALLP @RL2	CALLP @@RW5+d8	INCL RL2	INCL @RW5+d8	DECL RL2	DECL @RW5+d8	MOVL A,RL2	MOVL A, @RW5+d8	MOVL RL2,A	MOVL @R W5+d8,A	MOV R5,#8	MOV @R W5+d8,#8	MOVEA A,RW5	MOVEA A, @RW5+d8
+ 6	JMPP @RL3	JMPP @@RW6+d8	CALLP @RL3	CALLP @@RW6+d8	INCL RL3	INCL @RW6+d8	DECL RL3	DECL @RW6+d8	MOVL A,RL3	MOVL A, @RW6+d8	MOVL RL3,A	MOVL @R W6+d8,A	MOV R6,#8	MOV @R W6+d8,#8	MOVEA A,RW6	MOVEA A, @RW6+d8
+ 7	JMPP @RL3	JMPP @@RW7+d8	CALLP @RL3	CALLP @@RW7+d8	INCL RL3	INCL @RW7+d8	DECL RL3	DECL @RW7+d8	MOVL A,RL3	MOVL A, @RW7+d8	MOVL RL3,A	MOVL @R W7+d8,A	MOV R7,#8	MOV @R W7+d8,#8	MOVEA A,RW7	MOVEA A, @RW7+d8
+ 8	JMPP @@RW0	JMPP @ @RW0+d16	CALLP @@RW0	CALLP @ @RW0+d16	INCL @RW0	INCL @RW0+d16	DECL @RW0	DECL @RW0+d16	MOVL A,@RW0	MOVL A, @RW0+d16	MOVL @RW0,A	MOVL @R W0+d16,A	MOV @RW0,#8	MOV @RW 0+d16,#8	MOVEA A,@RW0	MOVEA A, @RW0+d16
+ 9	JMPP @@RW1	JMPP @ @RW1+d16	CALLP @@RW1	CALLP @ @RW1+d16	INCL @RW1	INCL @RW1+d16	DECL @RW1	DECL @RW1+d16	MOVL A,@RW1	MOVL A, @RW1+d16	MOVL @RW1,A	MOVL @R W1+d16,A	MOV @RW1,#8	MOV @RW 1+d16,#8	MOVEA A,@RW1	MOVEA A, @RW1+d16
+ A	JMPP @@RW2	JMPP @ @RW2+d16	CALLP @@RW2	CALLP @ @RW2+d16	INCL @RW2	INCL @RW2+d16	DECL @RW2	DECL @RW2+d16	MOVL A,@RW2	MOVL A, @RW2+d16	MOVL @RW2,A	MOVL @R W2+d16,A	MOV @RW2,#8	MOV @RW 2+d16,#8	MOVEA A,@RW2	MOVEA A, @RW2+d16
+ B	JMPP @@RW3	JMPP @ @RW3+d16	CALLP @@RW3	CALLP @ @RW3+d16	INCL @RW3	INCL @RW3+d16	DECL @RW3	DECL @RW3+d16	MOVL A,@RW3	MOVL A, @RW3+d16	MOVL @RW3,A	MOVL @R W3+d16,A	MOV @RW3,#8	MOV @RW 3+d16,#8	MOVEA A,@RW3	MOVEA A, @RW3+d16
+ C	JMPP @@RW0+	JMPP @ @RW0+RW7	CALLP @@RW0+	CALLP @ @RW0+RW7	INCL @RW0+	INCL @RW0+RW7	DECL @RW0+	DECL @RW0+RW7	MOVL A,@RW0+	MOVL A, @RW0+RW7	MOVL @RW0+,A	MOVL @R W0+RW7,A	MOV @RW0+,#8	MOV @RW 0+RW7,#8	MOVEA A,@RW0+	MOVEA A, @RW0+RW7
+ D	JMPP @@RW1+	JMPP @ @RW1+RW7	CALLP @@RW1+	CALLP @ @RW1+RW7	INCL @RW1+	INCL @RW1+RW7	DECL @RW1+	DECL @RW1+RW7	MOVL A,@RW1+	MOVL A, @RW1+RW7	MOVL @RW1+,A	MOVL @R W1+RW7,A	MOV @RW1+,#8	MOV @RW 1+RW7,#8	MOVEA A,@RW1+	MOVEA A, @RW1+RW7
+ E	JMPP @@RW2+	JMPP @@PC+d16	CALLP @@RW2+	CALLP @@PC+d16	INCL @RW2+	INCL @PC+d16	DECL @RW2+	DECL @PC+d16	MOVL A,@RW2+	MOVL A, @PC+d16	MOVL @RW2+,A	MOVL @P C+d16,A	MOV @RW2+,#8	MOV @P C+d16,#8	MOVEA A,@RW2+	MOVEA A, @PC+d16
+ F	JMPP @@RW3+	JMPP @addr16	CALLP @@RW3+	CALLP @addr16	INCL @RW3+	INCL addr16	DECL @RW3+	DECL addr16	MOVL A,@RW3+	MOVL A, addr16	MOVL @RW3+,A	MOVL addr16,A	MOV @RW3+,#8	MOV addr16,#8	MOVEA A,@RW3+	MOVEA A, addr16

Table B.9-8 ea Instruction 3 (first byte = 72_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ROLC R0	ROLC @RW0+d8	RORC R0	RORC @RW0+d8	INC R0	INC @RW0+d8	DEC R0	DEC @RW0+d8	MOV A,R0	MOV A, @RW0+d8	MOV R0,A	MOV @RW0+d8,A	MOVX A,R0	MOVX A, @RW0+d8	XCH A,R0	XCH A, @RW0+d8
+ 1	ROLC R1	ROLC @RW1+d8	RORC R1	RORC @RW1+d8	INC R1	INC @RW1+d8	DEC R1	DEC @RW1+d8	MOV A,R1	MOV A, @RW1+d8	MOV R1,A	MOV @RW1+d8,A	MOVX A,R1	MOVX A, @RW1+d8	XCH A,R1	XCH A, @RW1+d8
+ 2	ROLC R2	ROLC @RW2+d8	RORC R2	RORC @RW2+d8	INC R2	INC @RW2+d8	DEC R2	DEC @RW2+d8	MOV A,R2	MOV A, @RW2+d8	MOV R2,A	MOV @RW2+d8,A	MOVX A,R2	MOVX A, @RW2+d8	XCH A,R2	XCH A, @RW2+d8
+ 3	ROLC R3	ROLC @RW3+d8	RORC R3	RORC @RW3+d8	INC R3	INC @RW3+d8	DEC R3	DEC @RW3+d8	MOV A,R3	MOV A, @RW3+d8	MOV R3,A	MOV @RW3+d8,A	MOVX A,R3	MOVX A, @RW3+d8	XCH A,R3	XCH A, @RW3+d8
+ 4	ROLC R4	ROLC @RW4+d8	RORC R4	RORC @RW4+d8	INC R4	INC @RW4+d8	DEC R4	DEC @RW4+d8	MOV A,R4	MOV A, @RW4+d8	MOV R4,A	MOV @RW4+d8,A	MOVX A,R4	MOVX A, @RW4+d8	XCH A,R4	XCH A, @RW4+d8
+ 5	ROLC R5	ROLC @RW5+d8	RORC R5	RORC @RW5+d8	INC R5	INC @RW5+d8	DEC R5	DEC @RW5+d8	MOV A,R5	MOV A, @RW5+d8	MOV R5,A	MOV @RW5+d8,A	MOVX A,R5	MOVX A, @RW5+d8	XCH A,R5	XCH A, @RW5+d8
+ 6	ROLC R6	ROLC @RW6+d8	RORC R6	RORC @RW6+d8	INC R6	INC @RW6+d8	DEC R6	DEC @RW6+d8	MOV A,R6	MOV A, @RW6+d8	MOV R6,A	MOV @RW6+d8,A	MOVX A,R6	MOVX A, @RW6+d8	XCH A,R6	XCH A, @RW6+d8
+ 7	ROLC R7	ROLC @RW7+d8	RORC R7	RORC @RW7+d8	INC R7	INC @RW7+d8	DEC R7	DEC @RW7+d8	MOV A,R7	MOV A, @RW7+d8	MOV R7,A	MOV @RW7+d8,A	MOVX A,R7	MOVX A, @RW7+d8	XCH A,R7	XCH A, @RW7+d8
+ 8	ROLC @RW0	ROLC @RW0+d16	RORC @RW0	RORC @RW0+d16	INC @RW0	INC @RW0+d16	DEC @RW0	DEC @RW0+d16	MOV A,@RW0	MOV A, @RW0+d16	MOV @RW0,A	MOV @RW0+d16,A	MOVX A,@RW0	MOVX A, @RW0+d16	XCH A,@RW0	XCH A, @RW0+d16
+ 9	ROLC @RW1	ROLC @RW1+d16	RORC @RW1	RORC @RW1+d16	INC @RW1	INC @RW1+d16	DEC @RW1	DEC @RW1+d16	MOV A,@RW1	MOV A, @RW1+d16	MOV @RW1,A	MOV @RW1+d16,A	MOVX A,@RW1	MOVX A, @RW1+d16	XCH A,@RW1	XCH A, @RW1+d16
+ A	ROLC @RW2	ROLC @RW2+d16	RORC @RW2	RORC @RW2+d16	INC @RW2	INC @RW2+d16	DEC @RW2	DEC @RW2+d16	MOV A,@RW2	MOV A, @RW2+d16	MOV @RW2,A	MOV @RW2+d16,A	MOVX A,@RW2	MOVX A, @RW2+d16	XCH A,@RW2	XCH A, @RW2+d16
+ B	ROLC @RW3	ROLC @RW3+d16	RORC @RW3	RORC @RW3+d16	INC @RW3	INC @RW3+d16	DEC @RW3	DEC @RW3+d16	MOV A,@RW3	MOV A, @RW3+d16	MOV @RW3,A	MOV @RW3+d16,A	MOVX A,@RW3	MOVX A, @RW3+d16	XCH A,@RW3	XCH A, @RW3+d16
+ C	ROLC @RW0+	ROLC @RW0+RW7	RORC @RW0+	RORC @RW0+RW7	INC @RW0+	INC @RW0+RW7	DEC @RW0+	DEC @RW0+RW7	MOV A,@RW0+	MOV A, @RW0+RW7	MOV @RW0+,A	MOV @RW0+RW7,A	MOVX A,@RW0+	MOVX A, @RW0+RW7	XCH A,@RW0+	XCH A, @RW0+RW7
+ D	ROLC @RW1+	ROLC @RW1+RW7	RORC @RW1+	RORC @RW1+RW7	INC @RW1+	INC @RW1+RW7	DEC @RW1+	DEC @RW1+RW7	MOV A,@RW1+	MOV A, @RW1+RW7	MOV @RW1+,A	MOV @RW1+RW7,A	MOVX A,@RW1+	MOVX A, @RW1+RW7	XCH A,@RW1+	XCH A, @RW1+RW7
+ E	ROLC @RW2+	ROLC @PC+d16	RORC @RW2+	RORC @PC+d16	INC @RW2+	INC @PC+d16	DEC @RW2+	DEC @PC+d16	MOV A,@RW2+	MOV A, @PC+d16	MOV @RW2+,A	MOV @PC+d16,A	MOVX A,@RW2+	MOVX A, @PC+d16	XCH A,@RW2+	XCH A, @PC+d16
+ F	ROLC @RW3+	ROLC addr16	RORC @RW3+	RORC addr16	INC @RW3+	INC addr16	DEC @RW3+	DEC addr16	MOV A,@RW3+	MOV A, addr16	MOV @RW3+,A	MOV addr16,A	MOVX A,@RW3+	MOVX A, addr16	XCH A,@RW3+	XCH A, addr16

Table B.9-9 ea Instruction 4 (first byte = 73$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMP @RW0	JMP @@RW0+d8	CALL @RW0	CALL @@RW0+d8	INCW RW0	INCW @RW0+d8	DECW RW0	DECW @RW0+d8	MOVW A,RW0	MOVW A, @RW0+d8	MOVW RW0,A	MOVW @R W0+d8,A	MOVW RW0,#16	MOVW @RW 0+d8,#16	XCHW A,RW0	XCHW A, @RW0+d8
+ 1	JMP @RW1	JMP @@RW1+d8	CALL @RW1	CALL @@RW1+d8	INCW RW1	INCW @RW1+d8	DECW RW1	DECW @RW1+d8	MOVW A,RW1	MOVW A, @RW1+d8	MOVW RW1,A	MOVW @R W1+d8,A	MOVW RW1,#16	MOVW @RW 1+d8,#16	XCHW A,RW1	XCHW A, @RW1+d8
+ 2	JMP @RW2	JMP @@RW2+d8	CALL @RW2	CALL @@RW2+d8	INCW RW2	INCW @RW2+d8	DECW RW2	DECW @RW2+d8	MOVW A,RW2	MOVW A, @RW2+d8	MOVW RW2,A	MOVW @R W2+d8,A	MOVW RW2,#16	MOVW @RW 2+d8,#16	XCHW A,RW2	XCHW A, @RW2+d8
+ 3	JMP @RW3	JMP @@RW3+d8	CALL @RW3	CALL @@RW3+d8	INCW RW3	INCW @RW3+d8	DECW RW3	DECW @RW3+d8	MOVW A,RW3	MOVW A, @RW3+d8	MOVW RW3,A	MOVW @R W3+d8,A	MOVW RW3,#16	MOVW @RW 3+d8,#16	XCHW A,RW3	XCHW A, @RW3+d8
+ 4	JMP @RW4	JMP @@RW4+d8	CALL @RW4	CALL @@RW4+d8	INCW RW4	INCW @RW4+d8	DECW RW4	DECW @RW4+d8	MOVW A,RW4	MOVW A, @RW4+d8	MOVW RW4,A	MOVW @R W4+d8,A	MOVW RW4,#16	MOVW @RW 4+d8,#16	XCHW A,RW4	XCHW A, @RW4+d8
+ 5	JMP @RW5	JMP @@RW5+d8	CALL @RW5	CALL @@RW5+d8	INCW RW5	INCW @RW5+d8	DECW RW5	DECW @RW5+d8	MOVW A,RW5	MOVW A, @RW5+d8	MOVW RW5,A	MOVW @R W5+d8,A	MOVW RW5,#16	MOVW @RW 5+d8,#16	XCHW A,RW5	XCHW A, @RW5+d8
+ 6	JMP @RW6	JMP @@RW6+d8	CALL @RW6	CALL @@RW6+d8	INCW RW6	INCW @RW6+d8	DECW RW6	DECW @RW6+d8	MOVW A,RW6	MOVW A, @RW6+d8	MOVW RW6,A	MOVW @R W6+d8,A	MOVW RW6,#16	MOVW @RW 6+d8,#16	XCHW A,RW6	XCHW A, @RW6+d8
+ 7	JMP @RW7	JMP @@RW7+d8	CALL @RW7	CALL @@RW7+d8	INCW RW7	INCW @RW7+d8	DECW RW7	DECW @RW7+d8	MOVW A,RW7	MOVW A, @RW7+d8	MOVW RW7,A	MOVW @R W7+d8,A	MOVW RW7,#16	MOVW @RW 7+d8,#16	XCHW A,RW7	XCHW A, @RW7+d8
+ 8	JMP @@RW0	JMP @ @RW0+d16	CALL @@RW0	CALL @ @RW0+d16	INCW @RW0	INCW @RW0+d16	DECW @RW0	DECW @RW0+d16	MOVW A,@RW0	MOVW A, @RW0+d16	MOVW @RW0,A	MOVW @R W0+d16,A	MOVW @RW0,#16	MOVW@RW0 +d16,#16	XCHW A,@RW0	XCHW A, @RW0+d16
+ 9	JMP @@RW1	JMP @ @RW1+d16	CALL @@RW1	CALL @ @RW1+d16	INCW @RW1	INCW @RW1+d16	DECW @RW1	DECW @RW1+d16	MOVW A,@RW1	MOVW A, @RW1+d16	MOVW @RW1,A	MOVW @R W1+d16,A	MOVW @RW1,#16	MOVW@RW1 +d16,#16	XCHW A,@RW1	XCHW A, @RW1+d16
+ A	JMP @@RW2	JMP @ @RW2+d16	CALL @@RW2	CALL @ @RW2+d16	INCW @RW2	INCW @RW2+d16	DECW @RW2	DECW @RW2+d16	MOVW A,@RW2	MOVW A, @RW2+d16	MOVW @RW2,A	MOVW @R W2+d16,A	MOVW @RW2,#16	MOVW@RW2 +d16,#16	XCHW A,@RW2	XCHW A, @RW2+d16
+ B	JMP @@RW3	JMP @ @RW3+d16	CALL @@RW3	CALL @ @RW3+d16	INCW @RW3	INCW @RW3+d16	DECW @RW3	DECW @RW3+d16	MOVW A,@RW3	MOVW A, @RW3+d16	MOVW @RW3,A	MOVW @R W3+d16,A	MOVW @RW3,#16	MOVW@RW3 +d16,#16	XCHW A,@RW3	XCHW A, @RW3+d16
+ C	JMP @@RW0+	JMP @ @RW0+RW7	CALL @@RW0+	CALL @ @RW0+RW7	INCW @RW0+	INCW @RW0+RW7	DECW @RW0+	DECW @RW0+RW7	MOVW A,@RW0+	MOVW A, @RW0+RW7	MOVW @RW0+,A	MOVW @R W0+RW7,A	MOVW @ RW0+,#16	MOVW@RW0 +RW7,#16	XCHW A,@RW0+	XCHW A, @RW0+RW7
+ D	JMP @@RW1+	JMP @ @RW1+RW7	CALL @@RW1+	CALL @ @RW1+RW7	INCW @RW1+	INCW @RW1+RW7	DECW @RW1+	DECW @RW1+RW7	MOVW A,@RW1+	MOVW A, @RW1+RW7	MOVW @RW1+,A	MOVW @R W1+RW7,A	MOVW @ RW1+,#16	MOVW@RW1 +RW7,#16	XCHW A,@RW1+	XCHW A, @RW1+RW7
+ E	JMP @@RW2+	JMP @@PC+d16	CALL @@RW2+	CALL @@PC+d16	INCW @RW2+	INCW @PC+d16	DECW @RW2+	DECW @PC+d16	MOVW A,@RW2+	MOVW A, @PC+d16	MOVW @RW2+,A	MOVW @P C+d16,A	MOVW @ RW2+,#16	MOVW @PC +d16,#16	XCHW A,@RW2+	XCHW A, @PC+d16
+ F	JMP @@RW3+	JMP @addr16	CALL @@RW3+	CALL @addr16	INCW @RW3+	INCW addr16	DECW @RW3+	DECW addr16	MOVW A,@RW3+	MOVW A, addr16	MOVW @RW3+,A	MOVW addr16,A	MOVW @ RW3+,#16	MOVW addr16,#16	XCHW A,@RW3+	XCHW A, addr16

Table B.9-10 ea Instruction 5 (first byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A,R0	ADD A,@RW0+d8	SUB A,R0	SUB A,@RW0+d8	ADDC A,R0	ADDC A,@RW0+d8	CMP A,R0	CMP A,@RW0+d8	AND A,R0	AND A,@RW0+d8	OR A,R0	OR A,@RW0+d8	XOR A,R0	XOR A,@RW0+d8	DBNZ R0,r	DBNZ @RW0+d8,r
+1	ADD A,R1	ADD A,@RW1+d8	SUB A,R1	SUB A,@RW1+d8	ADDC A,R1	ADDC A,@RW1+d8	CMP A,R1	CMP A,@RW1+d8	AND A,R1	AND A,@RW1+d8	OR A,R1	OR A,@RW1+d8	XOR A,R1	XOR A,@RW1+d8	DBNZ R1,r	DBNZ @RW1+d8,r
+2	ADD A,R2	ADD A,@RW2+d8	SUB A,R2	SUB A,@RW2+d8	ADDC A,R2	ADDC A,@RW2+d8	CMP A,R2	CMP A,@RW2+d8	AND A,R2	AND A,@RW2+d8	OR A,R2	OR A,@RW2+d8	XOR A,R2	XOR A,@RW2+d8	DBNZ R2,r	DBNZ @RW2+d8,r
+3	ADD A,R3	ADD A,@RW3+d8	SUB A,R3	SUB A,@RW3+d8	ADDC A,R3	ADDC A,@RW3+d8	CMP A,R3	CMP A,@RW3+d8	AND A,R3	AND A,@RW3+d8	OR A,R3	OR A,@RW3+d8	XOR A,R3	XOR A,@RW3+d8	DBNZ R3,r	DBNZ @RW3+d8,r
+4	ADD A,R4	ADD A,@RW4+d8	SUB A,R4	SUB A,@RW4+d8	ADDC A,R4	ADDC A,@RW4+d8	CMP A,R4	CMP A,@RW4+d8	AND A,R4	AND A,@RW4+d8	OR A,R4	OR A,@RW4+d8	XOR A,R4	XOR A,@RW4+d8	DBNZ R4,r	DBNZ @RW4+d8,r
+5	ADD A,R5	ADD A,@RW5+d8	SUB A,R5	SUB A,@RW5+d8	ADDC A,R5	ADDC A,@RW5+d8	CMP A,R5	CMP A,@RW5+d8	AND A,R5	AND A,@RW5+d8	OR A,R5	OR A,@RW5+d8	XOR A,R5	XOR A,@RW5+d8	DBNZ R5,r	DBNZ @RW5+d8,r
+6	ADD A,R6	ADD A,@RW6+d8	SUB A,R6	SUB A,@RW6+d8	ADDC A,R6	ADDC A,@RW6+d8	CMP A,R6	CMP A,@RW6+d8	AND A,R6	AND A,@RW6+d8	OR A,R6	OR A,@RW6+d8	XOR A,R6	XOR A,@RW6+d8	DBNZ R6,r	DBNZ @RW6+d8,r
+7	ADD A,R7	ADD A,@RW7+d8	SUB A,R7	SUB A,@RW7+d8	ADDC A,R7	ADDC A,@RW7+d8	CMP A,R7	CMP A,@RW7+d8	AND A,R7	AND A,@RW7+d8	OR A,R7	OR A,@RW7+d8	XOR A,R7	XOR A,@RW7+d8	DBNZ R7,r	DBNZ @RW7+d8,r
+8	ADD A,@RW0	ADD A,@RW0+d16	SUB A,@RW0	SUB A,@RW0+d16	ADDC A,@RW0	ADDC A,@RW0+d16	CMP A,@RW0	CMP A,@RW0+d16	AND A,@RW0	AND A,@RW0+d16	OR A,@RW0	OR A,@RW0+d16	XOR A,@RW0	XOR A,@RW0+d16	DBNZ @RW0,r	DBNZ @RW0+d16,r
+9	ADD A,@RW1	ADD A,@RW1+d16	SUB A,@RW1	SUB A,@RW1+d16	ADDC A,@RW1	ADDC A,@RW1+d16	CMP A,@RW1	CMP A,@RW1+d16	AND A,@RW1	AND A,@RW1+d16	OR A,@RW1	OR A,@RW1+d16	XOR A,@RW1	XOR A,@RW1+d16	DBNZ @RW1,r	DBNZ @RW1+d16,r
+A	ADD A,@RW2	ADD A,@RW2+d16	SUB A,@RW2	SUB A,@RW2+d16	ADDC A,@RW2	ADDC A,@RW2+d16	CMP A,@RW2	CMP A,@RW2+d16	AND A,@RW2	AND A,@RW2+d16	OR A,@RW2	OR A,@RW2+d16	XOR A,@RW2	XOR A,@RW2+d16	DBNZ @RW2,r	DBNZ @RW2+d16,r
+B	ADD A,@RW3	ADD A,@RW3+d16	SUB A,@RW3	SUB A,@RW3+d16	ADDC A,@RW3	ADDC A,@RW3+d16	CMP A,@RW3	CMP A,@RW3+d16	AND A,@RW3	AND A,@RW3+d16	OR A,@RW3	OR A,@RW3+d16	XOR A,@RW3	XOR A,@RW3+d16	DBNZ @RW3,r	DBNZ @RW3+d16,r
+C	ADD A,@RW0+	ADD A,@RW0+RW7	SUB A,@RW0+	SUB A,@RW0+RW7	ADDC A,@RW0+	ADDC A,@RW0+RW7	CMP A,@RW0+	CMP A,@RW0+RW7	AND A,@RW0+	AND A,@RW0+RW7	OR A,@RW0+	OR A,@RW0+RW7	XOR A,@RW0+	XOR A,@RW0+RW7	DBNZ @RW0+,r	DBNZ @RW0+RW7,r
+D	ADD A,@RW1+	ADD A,@RW1+RW7	SUB A,@RW1+	SUB A,@RW1+RW7	ADDC A,@RW1+	ADDC A,@RW1+RW7	CMP A,@RW1+	CMP A,@RW1+RW7	AND A,@RW1+	AND A,@RW1+RW7	OR A,@RW1+	OR A,@RW1+RW7	XOR A,@RW1+	XOR A,@RW1+RW7	DBNZ @RW1+,r	DBNZ @RW1+RW7,r
+E	ADD A,@RW2+	ADD A,@PC+d16	SUB A,@RW2+	SUB A,@PC+d16	ADDC A,@RW2+	ADDC A,@PC+d16	CMP A,@RW2+	CMP A,@PC+d16	AND A,@RW2+	AND A,@PC+d16	OR A,@RW2+	OR A,@PC+d16	XOR A,@RW2+	XOR A,@PC+d16	DBNZ @RW2+,r	DBNZ @PC+d16,r
+F	ADD A,@RW3+	ADD A,addr16	SUB A,@RW3+	SUB A,addr16	ADDC A,@RW3+	ADDC A,addr16	CMP A,@RW3+	CMP A,addr16	AND A,@RW3+	AND A,addr16	OR A,@RW3+	OR A,addr16	XOR A,@RW3+	XOR A,addr16	DBNZ @RW3+,r	DBNZ addr16,r

Table B.9-11 ea Instruction 6 (first byte = 75_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADD R0,A	ADD @R W0+d8,A	SUB R0,A	SUB @R W0+d8,A	SUBC A,R0	SUBC A, @RW0+d8	NEG R0	NEG @RW0+d8	AND R0,A	AND @R W0+d8,A	OR R0,A	OR @R W0+d8,A	XOR R0,A	XOR @R W0+d8,A	NOT R0	NOT @RW0+d8
+ 1	ADD R1,A	ADD @R W1+d8,A	SUB R1,A	SUB @R W1+d8,A	SUBC A,R1	SUBC A, @RW1+d8	NEG R1	NEG @RW1+d8	AND R1,A	AND @R W1+d8,A	OR R1,A	OR @R W1+d8,A	XOR R1,A	XOR @R W1+d8,A	NOT R1	NOT @RW1+d8
+ 2	ADD R2,A	ADD @R W2+d8,A	SUB R2,A	SUB @R W2+d8,A	SUBC A,R2	SUBC A, @RW2+d8	NEG R2	NEG @RW2+d8	AND R2,A	AND @R W2+d8,A	OR R2,A	OR @R W2+d8,A	XOR R2,A	XOR @R W2+d8,A	NOT R2	NOT @RW2+d8
+ 3	ADD R3,A	ADD @R W3+d8,A	SUB R3,A	SUB @R W3+d8,A	SUBC A,R3	SUBC A, @RW3+d8	NEG R3	NEG @RW3+d8	AND R3,A	AND @R W3+d8,A	OR R3,A	OR @R W3+d8,A	XOR R3,A	XOR @R W3+d8,A	NOT R3	NOT @RW3+d8
+ 4	ADD R4,A	ADD @R W4+d8,A	SUB R4,A	SUB @R W4+d8,A	SUBC A,R4	SUBC A, @RW4+d8	NEG R4	NEG @RW4+d8	AND R4,A	AND @R W4+d8,A	OR R4,A	OR @R W4+d8,A	XOR R4,A	XOR @R W4+d8,A	NOT R4	NOT @RW4+d8
+ 5	ADD R5,A	ADD @R W5+d8,A	SUB R5,A	SUB @R W5+d8,A	SUBC A,R5	SUBC A, @RW5+d8	NEG R5	NEG @RW5+d8	AND R5,A	AND @R W5+d8,A	OR R5,A	OR @R W5+d8,A	XOR R5,A	XOR @R W5+d8,A	NOT R5	NOT @RW5+d8
+ 6	ADD R6,A	ADD @R W6+d8,A	SUB R6,A	SUB @R W6+d8,A	SUBC A,R6	SUBC A, @RW6+d8	NEG R6	NEG @RW6+d8	AND R6,A	AND @R W6+d8,A	OR R6,A	OR @R W6+d8,A	XOR R6,A	XOR @R W6+d8,A	NOT R6	NOT @RW6+d8
+ 7	ADD R7,A	ADD @R W7+d8,A	SUB R7,A	SUB @R W7+d8,A	SUBC A,R7	SUBC A, @RW7+d8	NEG R7	NEG @RW7+d8	AND R7,A	AND @R W7+d8,A	OR R7,A	OR @R W7+d8,A	XOR R7,A	XOR @R W7+d8,A	NOT R7	NOT @RW7+d8
+ 8	ADD @RW0,A	ADD @R W0+d16,A	SUB @RW0,A	SUB @R W0+d16,A	SUBC A,@RW0	SUBC A, @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0,A	AND @R W0+d16,A	OR @RW0,A	OR @R W0+d16,A	XOR @RW0,A	XOR @R W0+d16,A	NOT @RW0	NOT @RW0+d16
+ 9	ADD @RW1,A	ADD @R W1+d16,A	SUB @RW1,A	SUB @R W1+d16,A	SUBC A,@RW1	SUBC A, @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1,A	AND @R W1+d16,A	OR @RW1,A	OR @R W1+d16,A	XOR @RW1,A	XOR @R W1+d16,A	NOT @RW1	NOT @RW1+d16
+ A	ADD @RW2,A	ADD @R W2+d16,A	SUB @RW2,A	SUB @R W2+d16,A	SUBC A,@RW2	SUBC A, @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2,A	AND @R W2+d16,A	OR @RW2,A	OR @R W2+d16,A	XOR @RW2,A	XOR @R W2+d16,A	NOT @RW2	NOT @RW2+d16
+ B	ADD @RW3,A	ADD @R W3+d16,A	SUB @RW3,A	SUB @R W3+d16,A	SUBC A,@RW3	SUBC A, @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3,A	AND @R W3+d16,A	OR @RW3,A	OR @R W3+d16,A	XOR @RW3,A	XOR @R W3+d16,A	NOT @RW3	NOT @RW3+d16
+ C	ADD @RW0+,A	ADD @R W0+RW7,A	SUB @RW0+,A	SUB @R W0+RW7,A	SUBC A,@RW0+	SUBC A, @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+,A	AND @R W0+RW7,A	OR @RW0+,A	OR @R W0+RW7,A	XOR @RW0+,A	XOR @R W0+RW7,A	NOT @RW0+	NOT @RW0+RW7
+ D	ADD @RW1+,A	ADD @R W1+RW7,A	SUB @RW1+,A	SUB @R W1+RW7,A	SUBC A,@RW1+	SUBC A, @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+,A	AND @R W1+RW7,A	OR @RW1+,A	OR @R W1+RW7,A	XOR @RW1+,A	XOR @R W1+RW7,A	NOT @RW1+	NOT @RW1+RW7
+ E	ADD @RW2+,A	ADD @P C+d16,A	SUB @RW2+,A	SUB @P C+d16,A	SUBC A,@RW2+	SUBC A, @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+,A	AND @P C+d16,A	OR @RW2+,A	OR @P C+d16,A	XOR @RW2+,A	XOR @P C+d16,A	NOT @RW2+	NOT @PC+d16
+ F	ADD @RW3+,A	ADD addr16,A	SUB @RW3+,A	SUB addr16,A	SUBC A,@RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+,A	AND addr16,A	OR @RW3+,A	OR addr16,A	XOR @RW3+,A	XOR addr16,A	NOT @RW3+	NOT addr16

Table B.9-12 ea Instruction 7 (first byte = 76_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW A,RW0	ADDW A, @RW0+d8	SUBW A,RW0	SUBW A, @RW0+d8	ADDCW A,RW0	ADDCW A, @RW0+d8	CMPW A,RW0	CMPW A, @RW0+d8	ANDW A,RW0	ANDW A, @RW0+d8	ORW A,RW0	ORW A, @RW0+d8	XORW A,RW0	XORW A, @RW0+d8	DWBNZ RW0,r	DWBNZ @RW0+d8,r
+ 1	ADDW A,RW1	ADDW A, @RW1+d8	SUBW A,RW1	SUBW A, @RW1+d8	ADDCW A,RW1	ADDCW A, @RW1+d8	CMPW A,RW1	CMPW A, @RW1+d8	ANDW A,RW1	ANDW A, @RW1+d8	ORW A,RW1	ORW A, @RW1+d8	XORW A,RW1	XORW A, @RW1+d8	DWBNZ RW1,r	DWBNZ @RW1+d8,r
+ 2	ADDW A,RW2	ADDW A, @RW2+d8	SUBW A,RW2	SUBW A, @RW2+d8	ADDCW A,RW2	ADDCW A, @RW2+d8	CMPW A,RW2	CMPW A, @RW2+d8	ANDW A,RW2	ANDW A, @RW2+d8	ORW A,RW2	ORW A, @RW2+d8	XORW A,RW2	XORW A, @RW2+d8	DWBNZ RW2,r	DWBNZ @RW2+d8,r
+ 3	ADDW A,RW3	ADDW A, @RW3+d8	SUBW A,RW3	SUBW A, @RW3+d8	ADDCW A,RW3	ADDCW A, @RW3+d8	CMPW A,RW3	CMPW A, @RW3+d8	ANDW A,RW3	ANDW A, @RW3+d8	ORW A,RW3	ORW A, @RW3+d8	XORW A,RW3	XORW A, @RW3+d8	DWBNZ RW3,r	DWBNZ @RW3+d8,r
+ 4	ADDW A,RW4	ADDW A, @RW4+d8	SUBW A,RW4	SUBW A, @RW4+d8	ADDCW A,RW4	ADDCW A, @RW4+d8	CMPW A,RW4	CMPW A, @RW4+d8	ANDW A,RW4	ANDW A, @RW4+d8	ORW A,RW4	ORW A, @RW4+d8	XORW A,RW4	XORW A, @RW4+d8	DWBNZ RW4,r	DWBNZ @RW4+d8,r
+ 5	ADDW A,RW5	ADDW A, @RW5+d8	SUBW A,RW5	SUBW A, @RW5+d8	ADDCW A,RW5	ADDCW A, @RW5+d8	CMPW A,RW5	CMPW A, @RW5+d8	ANDW A,RW5	ANDW A, @RW5+d8	ORW A,RW5	ORW A, @RW5+d8	XORW A,RW5	XORW A, @RW5+d8	DWBNZ RW5,r	DWBNZ @RW5+d8,r
+ 6	ADDW A,RW6	ADDW A, @RW6+d8	SUBW A,RW6	SUBW A, @RW6+d8	ADDCW A,RW6	ADDCW A, @RW6+d8	CMPW A,RW6	CMPW A, @RW6+d8	ANDW A,RW6	ANDW A, @RW6+d8	ORW A,RW6	ORW A, @RW6+d8	XORW A,RW6	XORW A, @RW6+d8	DWBNZ RW6,r	DWBNZ @RW6+d8,r
+ 7	ADDW A,RW7	ADDW A, @RW7+d8	SUBW A,RW7	SUBW A, @RW7+d8	ADDCW A,RW7	ADDCW A, @RW7+d8	CMPW A,RW7	CMPW A, @RW7+d8	ANDW A,RW7	ANDW A, @RW7+d8	ORW A,RW7	ORW A, @RW7+d8	XORW A,RW7	XORW A, @RW7+d8	DWBNZ RW7,r	DWBNZ @RW7+d8,r
+ 8	ADDW A,@RW0	ADDW A, @RW0+d16	SUBW A,@RW0	SUBW A, @RW0+d16	ADDCW A,@RW0	ADDCW A, @RW0+d16	CMPW A,@RW0	CMPW A, @RW0+d16	ANDW A,@RW0	ANDW A, @RW0+d16	ORW A,@RW0	ORW A, @RW0+d16	XORW A,@RW0	XORW A, @RW0+d16	DWBNZ @RW0,r	DWBNZ @RW0+d16,r
+ 9	ADDW A,@RW1	ADDW A, @RW1+d16	SUBW A,@RW1	SUBW A, @RW1+d16	ADDCW A,@RW1	ADDCW A, @RW1+d16	CMPW A,@RW1	CMPW A, @RW1+d16	ANDW A,@RW1	ANDW A, @RW1+d16	ORW A,@RW1	ORW A, @RW1+d16	XORW A,@RW1	XORW A, @RW1+d16	DWBNZ @RW1,r	DWBNZ @RW1+d16,r
+ A	ADDW A,@RW2	ADDW A, @RW2+d16	SUBW A,@RW2	SUBW A, @RW2+d16	ADDCW A,@RW2	ADDCW A, @RW2+d16	CMPW A,@RW2	CMPW A, @RW2+d16	ANDW A,@RW2	ANDW A, @RW2+d16	ORW A,@RW2	ORW A, @RW2+d16	XORW A,@RW2	XORW A, @RW2+d16	DWBNZ @RW2,r	DWBNZ @RW2+d16,r
+ B	ADDW A,@RW3	ADDW A, @RW3+d16	SUBW A,@RW3	SUBW A, @RW3+d16	ADDCW A,@RW3	ADDCW A, @RW3+d16	CMPW A,@RW3	CMPW A, @RW3+d16	ANDW A,@RW3	ANDW A, @RW3+d16	ORW A,@RW3	ORW A, @RW3+d16	XORW A,@RW3	XORW A, @RW3+d16	DWBNZ @RW3,r	DWBNZ @RW3+d16,r
+ C	ADDW A,@RW0+	ADDW A, @RW0+RW7	SUBW A,@RW0+	SUBW A, @RW0+RW7	ADDCW A,@RW0+	ADDCW A, @RW0+RW7	CMPW A,@RW0+	CMPW A, @RW0+RW7	ANDW A,@RW0+	ANDW A, @RW0+RW7	ORW A,@RW0+	ORW A, @RW0+RW7	XORW A,@RW0+	XORW A, @RW0+RW7	DWBNZ @RW0+,r	DWBNZ @RW0+RW7,r
+ D	ADDW A,@RW1+	ADDW A, @RW1+RW7	SUBW A,@RW1+	SUBW A, @RW1+RW7	ADDCW A,@RW1+	ADDCW A, @RW1+RW7	CMPW A,@RW1+	CMPW A, @RW1+RW7	ANDW A,@RW1+	ANDW A, @RW1+RW7	ORW A,@RW1+	ORW A, @RW1+RW7	XORW A,@RW1+	XORW A, @RW1+RW7	DWBNZ @RW1+,r	DWBNZ @RW1+RW7,r
+ E	ADDW A,@RW2+	ADDW A, @PC+d16	SUBW A,@RW2+	SUBW A, @PC+d16	ADDCW A,@RW2+	ADDCW A, @PC+d16	CMPW A,@RW2+	CMPW A, @PC+d16	ANDW A,@RW2+	ANDW A, @PC+d16	ORW A,@RW2+	ORW A, @PC+d16	XORW A,@RW2+	XORW A, @PC+d16	DWBNZ @RW2+,r	DWBNZ @PC+d16,r
+ F	ADDW A,@RW3+	ADDW A, addr16	SUBW A,@RW3+	SUBW A, addr16	ADDCW A,@RW3+	ADDCW A, addr16	CMPW A,@RW3+	CMPW A, addr16	ANDW A,@RW3+	ANDW A, addr16	ORW A,@RW3+	ORW A, addr16	XORW A,@RW3+	XORW A, addr16	DWBNZ @RW3+,r	DWBNZ addr16,r

Table B.9-13 ea Instruction 8 (first byte = 77$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW RW0,A	ADDW @R W0+d8,A	SUBW RW0,A	SUBW @R W0+d8,A	SUBCW A,RW0	SUBCW A, @RW0+d8	NEGW RW0	NEGW @RW0+d8	ANDW RW0,A	ANDW @R W0+d8,A	ORW RW0,A	ORW @R W0+d8,A	XORW RW0,A	XORW @R W0+d8,A	NOTW RW0	NOTW @RW0+d8
+ 1	ADDW RW1,A	ADDW @R W1+d8,A	SUBW RW1,A	SUBW @R W1+d8,A	SUBCW A,RW1	SUBCW A, @RW1+d8	NEGW RW1	NEGW @RW1+d8	ANDW RW1,A	ANDW @R W1+d8,A	ORW RW1,A	ORW @R W1+d8,A	XORW RW1,A	XORW @R W1+d8,A	NOTW RW1	NOTW @RW1+d8
+ 2	ADDW RW2,A	ADDW @R W2+d8,A	SUBW RW2,A	SUBW @R W2+d8,A	SUBCW A,RW2	SUBCW A, @RW2+d8	NEGW RW2	NEGW @RW2+d8	ANDW RW2,A	ANDW @R W2+d8,A	ORW RW2,A	ORW @R W2+d8,A	XORW RW2,A	XORW @R W2+d8,A	NOTW RW2	NOTW @RW2+d8
+ 3	ADDW RW3,A	ADDW @R W3+d8,A	SUBW RW3,A	SUBW @R W3+d8,A	SUBCW A,RW3	SUBCW A, @RW3+d8	NEGW RW3	NEGW @RW3+d8	ANDW RW3,A	ANDW @R W3+d8,A	ORW RW3,A	ORW @R W3+d8,A	XORW RW3,A	XORW @R W3+d8,A	NOTW RW3	NOTW @RW3+d8
+ 4	ADDW RW4,A	ADDW @R W4+d8,A	SUBW RW4,A	SUBW @R W4+d8,A	SUBCW A,RW4	SUBCW A, @RW4+d8	NEGW RW4	NEGW @RW4+d8	ANDW RW4,A	ANDW @R W4+d8,A	ORW RW4,A	ORW @R W4+d8,A	XORW RW4,A	XORW @R W4+d8,A	NOTW RW4	NOTW @RW4+d8
+ 5	ADDW RW5,A	ADDW @R W5+d8,A	SUBW RW5,A	SUBW @R W5+d8,A	SUBCW A,RW5	SUBCW A, @RW5+d8	NEGW RW5	NEGW @RW5+d8	ANDW RW5,A	ANDW @R W5+d8,A	ORW RW5,A	ORW @R W5+d8,A	XORW RW5,A	XORW @R W5+d8,A	NOTW RW5	NOTW @RW5+d8
+ 6	ADDW RW6,A	ADDW @R W6+d8,A	SUBW RW6,A	SUBW @R W6+d8,A	SUBCW A,RW6	SUBCW A, @RW6+d8	NEGW RW6	NEGW @RW6+d8	ANDW RW6,A	ANDW @R W6+d8,A	ORW RW6,A	ORW @R W6+d8,A	XORW RW6,A	XORW @R W6+d8,A	NOTW RW6	NOTW @RW6+d8
+ 7	ADDW RW7,A	ADDW @R W7+d8,A	SUBW RW7,A	SUBW @R W7+d8,A	SUBCW A,RW7	SUBCW A, @RW7+d8	NEGW RW7	NEGW @RW7+d8	ANDW RW7,A	ANDW @R W7+d8,A	ORW RW7,A	ORW @R W7+d8,A	XORW RW7,A	XORW @R W7+d8,A	NOTW RW7	NOTW @RW7+d8
+ 8	ADDW @RW0,A	ADDW @R W0+d16,A	SUBW @RW0,A	SUBW @R W0+d16,A	SUBCW A,@RW0	SUBCW A, @RW0+d16	NEGW @RW0	NEGW @RW0+d16	ANDW @RW0,A	ANDW @R W0+d16,A	ORW @RW0,A	ORW @R W0+d16,A	XORW @RW0,A	XORW @R W0+d16,A	NOTW @RW0	NOTW @RW0+d16
+ 9	ADDW @RW1,A	ADDW @R W1+d16,A	SUBW @RW1,A	SUBW @R W1+d16,A	SUBCW A,@RW1	SUBCW A, @RW1+d16	NEGW @RW1	NEGW @RW1+d16	ANDW @RW1,A	ANDW @R W1+d16,A	ORW @RW1,A	ORW @R W1+d16,A	XORW @RW1,A	XORW @R W1+d16,A	NOTW @RW1	NOTW @RW1+d16
+ A	ADDW @RW2,A	ADDW @R W2+d16,A	SUBW @RW2,A	SUBW @R W2+d16,A	SUBCW A,@RW2	SUBCW A, @RW2+d16	NEGW @RW2	NEGW @RW2+d16	ANDW @RW2,A	ANDW @R W2+d16,A	ORW @RW2,A	ORW @R W2+d16,A	XORW @RW2,A	XORW @R W2+d16,A	NOTW @RW2	NOTW @RW2+d16
+ B	ADDW @RW3,A	ADDW @R W3+d16,A	SUBW @RW3,A	SUBW @R W3+d16,A	SUBCW A,@RW3	SUBCW A, @RW3+d16	NEGW @RW3	NEGW @RW3+d16	ANDW @RW3,A	ANDW @R W3+d16,A	ORW @RW3,A	ORW @R W3+d16,A	XORW @RW3,A	XORW @R W3+d16,A	NOTW @RW3	NOTW @RW3+d16
+ C	ADDW @RW0+,A	ADDW @R W0+RW7,A	SUBW @RW0+,A	SUBW @R W0+RW7,A	SUBCW A,@RW0+	SUBCW A, @RW0+RW7	NEGW @RW0+	NEGW @RW0+RW7	ANDW @RW0+,A	ANDW @R W0+RW7,A	ORW @RW0+,A	ORW @R W0+RW7,A	XORW @RW0+,A	XORW @R W0+RW7,A	NOTW @RW0+	NOTW @RW0+RW7
+ D	ADDW @RW1+,A	ADDW @R W1+RW7,A	SUBW @RW1+,A	SUBW @R W1+RW7,A	SUBCW A,@RW1+	SUBCW A, @RW1+RW7	NEGW @RW1+	NEGW @RW1+RW7	ANDW @RW1+,A	ANDW @R W1+RW7,A	ORW @RW1+,A	ORW @R W1+RW7,A	XORW @RW1+,A	XORW @R W1+RW7,A	NOTW @RW1+	NOTW @RW1+RW7
+ E	ADDW @RW2+,A	ADDW @P C+d16,A	SUBW @RW2+,A	SUBW @P C+d16,A	SUBCW A,@RW2+	SUBCW A, @PC+d16	NEGW @RW2+	NEGW @PC+d16	ANDW @RW2+,A	ANDW @P C+d16,A	ORW @RW2+,A	ORW @P C+d16,A	XORW @RW2+,A	XORW @P C+d16,A	NOTW @RW2+	NOTW @PC+d16
+ F	ADDW @RW3+,A	ADDW addr16,A	SUBW @RW3+,A	SUBW addr16,A	SUBCW A,@RW3+	SUBCW A, addr16	NEGW @RW3+	NEGW addr16	ANDW @RW3+,A	ANDW addr16,A	ORW @RW3+,A	ORW addr16,A	XORW @RW3+,A	XORW addr16,A	NOTW @RW3+	NOTW addr16

Table B.9-14 ea Instruction 9 (first byte = 78_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MULU A,R0	MULU A, @RW0+d8	MULUW A,RW0	MULUW A, @RW0+d8	MUL A R0	MUL A, @RW0+d8	MULW A,RW0	MULW A, @RW0+d8	DIVU A,R0	DIVU A, @RW0+d8	DIVUW A,RW0	DIVUW A, @RW0+d8	DIV A,R0	DIV A, @RW0+d8	DIVW A,RW0	DIVW A, @RW0+d8
+ 1	MULU A,R1	MULU A, @RW1+d8	MULUW A,RW1	MULUW A, @RW1+d8	MUL A,R1	MUL A, @RW1+d8	MULW A,RW1	MULW A, @RW1+d8	DIVU A,R1	DIVU A, @RW1+d8	DIVUW A,RW1	DIVUW A, @RW1+d8	DIV A,R1	DIV A, @RW1+d8	DIVW A,RW1	DIVW A, @RW1+d8
+ 2	MULU A,R2	MULU A, @RW2+d8	MULUW A,RW2	MULUW A, @RW2+d8	MUL A,R2	MUL A, @RW2+d8	MULW A RW2	MULW A, @RW2+d8	DIVU A,R2	DIVU A, @RW2+d8	DIVUW A,RW2	DIVUW A, @RW2+d8	DIV A,R2	DIV A, @RW2+d8	DIVW A,RW2	DIVW A, @RW2+d8
+ 3	MULU A,R3	MULU A, @RW3+d8	MULUW A,RW3	MULUW A, @RW3+d8	MUL A,R3	MUL A, @RW3+d8	MULW A RW3	MULW A, @RW3+d8	DIVU A,R3	DIVU A, @RW3+d8	DIVUW A,RW3	DIVUW A, @RW3+d8	DIV A,R3	DIV A, @RW3+d8	DIVW A,RW3	DIVW A, @RW3+d8
+ 4	MULU A,R4	MULU A, @RW4+d8	MULUW A,RW4	MULUW A, @RW4+d8	MUL A,R4	MUL A, @RW4+d8	MULW A RW4	MULW A, @RW4+d8	DIVU A,R4	DIVU A, @RW4+d8	DIVUW A,RW4	DIVUW A, @RW4+d8	DIV A,R4	DIV A, @RW4+d8	DIVW A,RW4	DIVW A, @RW4+d8
+ 5	MULU A,R5	MULU A, @RW5+d8	MULUW A,RW5	MULUW A, @RW5+d8	MUL A,R5	MUL A, @RW5+d8	MULW A RW5	MULW A, @RW5+d8	DIVU A,R5	DIVU A, @RW5+d8	DIVUW A,RW5	DIVUW A, @RW5+d8	DIV A,R5	DIV A, @RW5+d8	DIVW A,RW5	DIVW A, @RW5+d8
+ 6	MULU A,R6	MULU A, @RW6+d8	MULUW A,RW6	MULUW A, @RW6+d8	MUL A R6	MUL A, @RW6+d8	MULW A,RW6	MULW A, @RW6+d8	DIVU A,R6	DIVU A, @RW6+d8	DIVUW A,RW6	DIVUW A, @RW6+d8	DIV A,R6	DIV A, @RW6+d8	DIVW A,RW6	DIVW A, @RW6+d8
+ 7	MULU A,R7	MULU A, @RW7+d8	MULUW A,RW7	MULUW A, @RW7+d8	MUL A,R7	MUL A, @RW7+d8	MULW A,RW7	MULW A, @RW7+d8	DIVU A,R7	DIVU A, @RW7+d8	DIVUW A,RW7	DIVUW A, @RW7+d8	DIV A,R7	DIV A, @RW7+d8	DIVW A,RW7	DIVW A, @RW7+d8
+ 8	MULU A,@RW0	MULU A, @RW0+d16	MULUW A,@RW0	MULUW A, @RW0+d16	MUL A,@RW0	MUL A, @RW0+d16	MULW A,@RW0	MULW A, @RW0+d16	DIVU A,@RW0	DIVU A, @RW0+d16	DIVUW A,@RW0	DIVUW A, @RW0+d16	DIV A,@RW0	DIV A, @RW0+d16	DIVW A,@RW0	DIVW A, @RW0+d16
+ 9	MULU A,@RW1	MULU A, @RW1+d16	MULUW A,@RW1	MULUW A, @RW1+d16	MUL A,@RW1	MUL A, @RW1+d16	MULW A @RW1	MULW A, @RW1+d16	DIVU A,@RW1	DIVU A, @RW1+d16	DIVUW A,@RW1	DIVUW A, @RW1+d16	DIV A,@RW1	DIV A, @RW1+d16	DIVW A,@RW1	DIVW A, @RW1+d16
+ A	MULU A,@RW2	MULU A, @RW2+d16	MULUW A,@RW2	MULUW A, @RW2+d16	MUL A,@RW2	MUL A, @RW2+d16	MULW A,@RW2	MULW A, @RW2+d16	DIVU A,@RW2	DIVU A, @RW2+d16	DIVUW A,@RW2	DIVUW A, @RW2+d16	DIV A,@RW2	DIV A, @RW2+d16	DIVW A,@RW2	DIVW A, @RW2+d16
+ B	MULU A,@RW3	MULU A, @RW3+d16	MULUW A,@RW3	MULUW A, @RW3+d16	MUL A,@RW3	MUL A, @RW3+d16	MULW A,@RW3	MULW A, @RW3+d16	DIVU A,@RW3	DIVU A, @RW3+d16	DIVUW A,@RW3	DIVUW A, @RW3+d16	DIV A,@RW3	DIV A, @RW3+d16	DIVW A,@RW3	DIVW A, @RW3+d16
+ C	MULU A,@RW0+	MULU A, @RW0+RW7	MULUW A,@RW0+	MULUW A, @RW0+RW7	MUL A,@RW0+	MUL A, @RW0+RW7	MULW A,@RW0+	MULW A, @RW0+RW7	DIVU A,@RW0+	DIVU A, @RW0+RW7	DIVUW A,@RW0+	DIVUW A, @RW0+RW7	DIV A,@RW0+	DIV A, @RW0+RW7	DIVW A,@RW0+	DIVW A, @RW0+RW7
+ D	MULU A,@RW1+	MULU A, @RW1+RW7	MULUW A,@RW1+	MULUW A, @RW1+RW7	MUL A,@RW1+	MUL A, @RW1+RW7	MULW A,@RW1+	MULW A, @RW1+RW7	DIVU A,@RW1+	DIVU A, @RW1+RW7	DIVUW A,@RW1+	DIVUW A, @RW1+RW7	DIV A,@RW1+	DIV A, @RW1+RW7	DIVW A,@RW1+	DIVW A, @RW1+RW7
+ E	MULU A,@RW2+	MULU A, @PC+d16	MULUW A,@RW2+	MULUW A, @PC+d16	MUL A @RW2+	MUL A, @PC+d16	MULW A,@RW2+	MULW A, @PC+d16	DIVU A,@RW2+	DIVU A, @PC+d16	DIVUW A,@RW2+	DIVUW A, @PC+d16	DIV A,@RW2+	DIV A, @PC+d16	DIVW A,@RW2+	DIVW A, @PC+d16
+ F	MULU A,@RW3+	MULU A, addr16	MULUW A,@RW3+	MULUW A, addr16	MUL A,@RW3+	MUL A, addr16	MULW A,@RW3+	MULW A, addr16	DIVU A,@RW3+	DIVU A, addr16	DIVUW A,@RW3+	DIVUW A, addr16	DIV A,@RW3+	DIV A, addr16	DIVW A,@RW3+	DIVW A, addr16

Table B.9-15 MOVEA RWi, ea Instruction (first byte = 79_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVEA RW0,RW0	MOVEA RW0 ,@RW0+d8	MOVEA RW1,RW0	MOVEA RW1 ,@RW0+d8	MOVEA RW2,RW0	MOVEA RW2 ,@RW0+d8	MOVEA RW3,RW0	MOVEA RW3 ,@RW0+d8	MOVEA RW4,RW0	MOVEA RW4 ,@RW0+d8	MOVEA RW5,RW0	MOVEA RW5 ,@RW0+d8	MOVEA RW6,RW0	MOVEA RW6 ,@RW0+d8	MOVEA RW7,RW0	MOVEA RW7 ,@RW0+d8
+ 1	MOVEA RW0,RW1	MOVEA RW0 ,@RW1+d8	MOVEA RW1,RW1	MOVEA RW1 ,@RW1+d8	MOVEA RW2,RW1	MOVEA RW2 ,@RW1+d8	MOVEA RW3,RW1	MOVEA RW3 ,@RW1+d8	MOVEA RW4,RW1	MOVEA RW4 ,@RW1+d8	MOVEA RW5,RW1	MOVEA RW5 ,@RW1+d8	MOVEA RW6,RW1	MOVEA RW6 ,@RW1+d8	MOVEA RW7,RW1	MOVEA RW7 ,@RW1+d8
+ 2	MOVEA RW0,RW2	MOVEA RW0 ,@RW2+d8	MOVEA RW1,RW2	MOVEA RW1 ,@RW2+d8	MOVEA RW2,RW2	MOVEA RW2 ,@RW2+d8	MOVEA RW3,RW2	MOVEA RW3 ,@RW2+d8	MOVEA RW4,RW2	MOVEA RW4 ,@RW2+d8	MOVEA RW5,RW2	MOVEA RW5 ,@RW2+d8	MOVEA RW6,RW2	MOVEA RW6 ,@RW2+d8	MOVEA RW7,RW2	MOVEA RW7 ,@RW2+d8
+ 3	MOVEA RW0,RW3	MOVEA RW0 ,@RW3+d8	MOVEA RW1,RW3	MOVEA RW1 ,@RW3+d8	MOVEA RW2,RW3	MOVEA RW2 ,@RW3+d8	MOVEA RW3,RW3	MOVEA RW3 ,@RW3+d8	MOVEA RW4,RW3	MOVEA RW4 ,@RW3+d8	MOVEA RW5,RW3	MOVEA RW5 ,@RW3+d8	MOVEA RW6,RW3	MOVEA RW6 ,@RW3+d8	MOVEA RW7,RW3	MOVEA RW7 ,@RW3+d8
+ 4	MOVEA RW0,RW4	MOVEA RW0 ,@RW4+d8	MOVEA RW1,RW4	MOVEA RW1 ,@RW4+d8	MOVEA RW2,RW4	MOVEA RW2 ,@RW4+d8	MOVEA RW3,RW4	MOVEA RW3 ,@RW4+d8	MOVEA RW4,RW4	MOVEA RW4 ,@RW4+d8	MOVEA RW5,RW4	MOVEA RW5 ,@RW4+d8	MOVEA RW6,RW4	MOVEA RW6 ,@RW4+d8	MOVEA RW7,RW4	MOVEA RW7 ,@RW4+d8
+ 5	MOVEA RW0,RW5	MOVEA RW0 ,@RW5+d8	MOVEA RW1,RW5	MOVEA RW1 ,@RW5+d8	MOVEA RW2,RW5	MOVEA RW2 ,@RW5+d8	MOVEA RW3,RW5	MOVEA RW3 ,@RW5+d8	MOVEA RW4,RW5	MOVEA RW4 ,@RW5+d8	MOVEA RW5,RW5	MOVEA RW5 ,@RW5+d8	MOVEA RW6,RW5	MOVEA RW6 ,@RW5+d8	MOVEA RW7,RW5	MOVEA RW7 ,@RW5+d8
+ 6	MOVEA RW0,RW6	MOVEA RW0 ,@RW6+d8	MOVEA RW1,RW6	MOVEA RW1 ,@RW6+d8	MOVEA RW2,RW6	MOVEA RW2 ,@RW6+d8	MOVEA RW3,RW6	MOVEA RW3 ,@RW6+d8	MOVEA RW4,RW6	MOVEA RW4 ,@RW6+d8	MOVEA RW5,RW6	MOVEA RW5 ,@RW6+d8	MOVEA RW6,RW6	MOVEA RW6 ,@RW6+d8	MOVEA RW7,RW6	MOVEA RW7 ,@RW6+d8
+ 7	MOVEA RW0,RW7	MOVEA RW0 ,@RW7+d8	MOVEA RW1,RW7	MOVEA RW1 ,@RW7+d8	MOVEA RW2,RW7	MOVEA RW2 ,@RW7+d8	MOVEA RW3,RW7	MOVEA RW3 ,@RW7+d8	MOVEA RW4,RW7	MOVEA RW4 ,@RW7+d8	MOVEA RW5,RW7	MOVEA RW5 ,@RW7+d8	MOVEA RW6,RW7	MOVEA RW6 ,@RW7+d8	MOVEA RW7,RW7	MOVEA RW7 ,@RW7+d8
+ 8	MOVEA RW0,@RW0	MOVEA RW0 ,@RW0+d16	MOVEA RW1,@RW0	MOVEA RW1 ,@RW0+d16	MOVEA RW2,@RW0	MOVEA RW2 ,@RW0+d16	MOVEA RW3,@RW0	MOVEA RW3 ,@RW0+d16	MOVEA RW4,@RW0	MOVEA RW4 ,@RW0+d16	MOVEA RW5,@RW0	MOVEA RW5 ,@RW0+d16	MOVEA RW6,@RW0	MOVEA RW6 ,@RW0+d16	MOVEA RW7,@RW0	MOVEA RW7 ,@RW0+d16
+ 9	MOVEA RW0,@RW1	MOVEA RW0 ,@RW1+d16	MOVEA RW1,@RW1	MOVEA RW1 ,@RW1+d16	MOVEA RW2,@RW1	MOVEA RW2 ,@RW1+d16	MOVEA RW3,@RW1	MOVEA RW3 ,@RW1+d16	MOVEA RW4,@RW1	MOVEA RW4 ,@RW1+d16	MOVEA RW5,@RW1	MOVEA RW5 ,@RW1+d16	MOVEA RW6,@RW1	MOVEA RW6 ,@RW1+d16	MOVEA RW7,@RW1	MOVEA RW7 ,@RW1+d16
+ A	MOVEA RW0,@RW2	MOVEA RW0 ,@RW2+d16	MOVEA RW1,@RW2	MOVEA RW1 ,@RW2+d16	MOVEA RW2,@RW2	MOVEA RW2 ,@RW2+d16	MOVEA RW3,@RW2	MOVEA RW3 ,@RW2+d16	MOVEA RW4,@RW2	MOVEA RW4 ,@RW2+d16	MOVEA RW5,@RW2	MOVEA RW5 ,@RW2+d16	MOVEA RW6,@RW2	MOVEA RW6 ,@RW2+d16	MOVEA RW7,@RW2	MOVEA RW7 ,@RW2+d16
+ B	MOVEA RW0,@RW3	MOVEA RW0 ,@RW3+d16	MOVEA RW1,@RW3	MOVEA RW1 ,@RW3+d16	MOVEA RW2,@RW3	MOVEA RW2 ,@RW3+d16	MOVEA RW3,@RW3	MOVEA RW3 ,@RW3+d16	MOVEA RW4,@RW3	MOVEA RW4 ,@RW3+d16	MOVEA RW5,@RW3	MOVEA RW5 ,@RW3+d16	MOVEA RW6,@RW3	MOVEA RW6 ,@RW3+d16	MOVEA RW7,@RW3	MOVEA RW7 ,@RW3+d16
+ C	MOVEA R W0,@RW0+	MOVEA RW0 ,@RW0+RW7	MOVEA R W1,@RW0+	MOVEA RW1 ,@RW0+RW7	MOVEA R W2,@RW0+	MOVEA RW2 ,@RW0+RW7	MOVEA R W3,@RW0+	MOVEA RW3 ,@RW0+RW7	MOVEA R W4,@RW0+	MOVEA RW4 ,@RW0+RW7	MOVEA R W5,@RW0+	MOVEA RW5 ,@RW0+RW7	MOVEA R W6,@RW0+	MOVEA RW6 ,@RW0+RW7	MOVEA R W7,@RW0+	MOVEA RW7 ,@RW0+RW7
+ D	MOVEA R W0,@RW1+	MOVEA RW0 ,@RW1+RW7	MOVEA R W1,@RW1+	MOVEA RW1 ,@RW1+RW7	MOVEA R W2,@RW1+	MOVEA RW2 ,@RW1+RW7	MOVEA R W3,@RW1+	MOVEA RW3 ,@RW1+RW7	MOVEA R W4,@RW1+	MOVEA RW4 ,@RW1+RW7	MOVEA R W5,@RW1+	MOVEA RW5 ,@RW1+RW7	MOVEA R W6,@RW1+	MOVEA RW6 ,@RW1+RW7	MOVEA R W7,@RW1+	MOVEA RW7 ,@RW1+RW7
+ E	MOVEA R W0,@RW2+	MOVEA RW0 ,@PC+d16	MOVEA R W1,@RW2+	MOVEA RW1 ,@PC+d16	MOVEA R W2,@RW2+	MOVEA RW2 ,@PC+d16	MOVEA R W3,@RW2+	MOVEA RW3 ,@PC+d16	MOVEA R W4,@RW2+	MOVEA RW4 ,@PC+d16	MOVEA R W5,@RW2+	MOVEA RW5 ,@PC+d16	MOVEA R W6,@RW2+	MOVEA RW6 ,@PC+d16	MOVEA R W7,@RW2+	MOVEA RW7 ,@PC+d16
+ F	MOVEA R W0,@RW3+	MOVEA RW0 ,addr16	MOVEA R W1,@RW3+	MOVEA RW1 ,addr16	MOVEA R W2,@RW3+	MOVEA RW2 ,addr16	MOVEA R W3,@RW3+	MOVEA RW3 ,addr16	MOVEA R W4,@RW3+	MOVEA RW4 ,addr16	MOVEA R W5,@RW3+	MOVEA RW5 ,addr16	MOVEA R W6,@RW3+	MOVEA RW6 ,addr16	MOVEA R W7,@RW3+	MOVEA RW7 ,addr16

Table B.9-16 MOV Ri, ea Instruction (first byte = 7A$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0,R0	MOV R0, @RW0+d8	MOV R1,R0	MOV R1, @RW0+d8	MOV R2,R0	MOV R2, @RW0+d8	MOV R3,R0	MOV R3, @RW0+d8	MOV R4,R0	MOV R4, @RW0+d8	MOV R5,R0	MOV R5, @RW0+d8	MOV R6,R0	MOV R6, @RW0+d8	MOV R7,R0	MOV R7, @RW0+d8
+1	MOV R0,R1	MOV R0, @RW1+d8	MOV R1,R1	MOV R1, @RW1+d8	MOV R2,R1	MOV R2, @RW1+d8	MOV R3,R1	MOV R3, @RW1+d8	MOV R4,R1	MOV R4, @RW1+d8	MOV R5,R1	MOV R5, @RW1+d8	MOV R6,R1	MOV R6, @RW1+d8	MOV R7,R1	MOV R7, @RW1+d8
+2	MOV R0,R2	MOV R0, @RW2+d8	MOV R1,R2	MOV R1, @RW2+d8	MOV R2,R2	MOV R2, @RW2+d8	MOV R3,R2	MOV R3, @RW2+d8	MOV R4,R2	MOV R4, @RW2+d8	MOV R5,R2	MOV R5, @RW2+d8	MOV R6,R2	MOV R6, @RW2+d8	MOV R7,R2	MOV R7, @RW2+d8
+3	MOV R0,R3	MOV R0, @RW3+d8	MOV R1,R3	MOV R1, @RW3+d8	MOV R2,R3	MOV R2, @RW3+d8	MOV R3,R3	MOV R3, @RW3+d8	MOV R4,R3	MOV R4, @RW3+d8	MOV R5,R3	MOV R5, @RW3+d8	MOV R6,R3	MOV R6, @RW3+d8	MOV R7,R3	MOV R7, @RW3+d8
+4	MOV R0,R4	MOV R0, @RW4+d8	MOV R1,R4	MOV R1, @RW4+d8	MOV R2,R4	MOV R2, @RW4+d8	MOV R3,R4	MOV R3, @RW4+d8	MOV R4,R4	MOV R4, @RW4+d8	MOV R5,R4	MOV R5, @RW4+d8	MOV R6,R4	MOV R6, @RW4+d8	MOV R7,R4	MOV R7, @RW4+d8
+5	MOV R0,R5	MOV R0, @RW5+d8	MOV R1,R5	MOV R1, @RW5+d8	MOV R2,R5	MOV R2, @RW5+d8	MOV R3,R5	MOV R3, @RW5+d8	MOV R4,R5	MOV R4, @RW5+d8	MOV R5,R5	MOV R5, @RW5+d8	MOV R6,R5	MOV R6, @RW5+d8	MOV R7,R5	MOV R7, @RW5+d8
+6	MOV R0,R6	MOV R0, @RW6+d8	MOV R1,R6	MOV R1, @RW6+d8	MOV R2,R6	MOV R2, @RW6+d8	MOV R3,R6	MOV R3, @RW6+d8	MOV R4,R6	MOV R4, @RW6+d8	MOV R5,R6	MOV R5, @RW6+d8	MOV R6,R6	MOV R6, @RW6+d8	MOV R7,R6	MOV R7, @RW6+d8
+7	MOV R0,R7	MOV R0, @RW7+d8	MOV R1,R7	MOV R1, @RW7+d8	MOV R2,R7	MOV R2, @RW7+d8	MOV R3,R7	MOV R3, @RW7+d8	MOV R4,R7	MOV R4, @RW7+d8	MOV R5,R7	MOV R5, @RW7+d8	MOV R6,R7	MOV R6, @RW7+d8	MOV R7,R7	MOV R7, @RW7+d8
+8	MOV R0,@RW0	MOV R0, @RW0+d16	MOV R1,@RW0	MOV R1, @RW0+d16	MOV R2,@RW0	MOV R2, @RW0+d16	MOV R3,@RW0	MOV R3, @RW0+d16	MOV R4,@RW0	MOV R4, @RW0+d16	MOV R5,@RW0	MOV R5, @RW0+d16	MOV R6,@RW0	MOV R6, @RW0+d16	MOV R7,@RW0	MOV R7, @RW0+d16
+9	MOV R0,@RW1	MOV R0, @RW1+d16	MOV R1,@RW1	MOV R1, @RW1+d16	MOV R2,@RW1	MOV R2, @RW1+d16	MOV R3,@RW1	MOV R3, @RW1+d16	MOV R4,@RW1	MOV R4, @RW1+d16	MOV R5,@RW1	MOV R5, @RW1+d16	MOV R6,@RW1	MOV R6, @RW1+d16	MOV R7,@RW1	MOV R7, @RW1+d16
+A	MOV R0,@RW2	MOV R0, @RW2+d16	MOV R1,@RW2	MOV R1, @RW2+d16	MOV R2,@RW2	MOV R2, @RW2+d16	MOV R3,@RW2	MOV R3, @RW2+d16	MOV R4,@RW2	MOV R4, @RW2+d16	MOV R5,@RW2	MOV R5, @RW2+d16	MOV R6,@RW2	MOV R6, @RW2+d16	MOV R7,@RW2	MOV R7, @RW2+d16
+B	MOV R0,@RW3	MOV R0, @RW3+d16	MOV R1,@RW3	MOV R1, @RW3+d16	MOV R2,@RW3	MOV R2, @RW3+d16	MOV R3,@RW3	MOV R3, @RW3+d16	MOV R4,@RW3	MOV R4, @RW3+d16	MOV R5,@RW3	MOV R5, @RW3+d16	MOV R6,@RW3	MOV R6, @RW3+d16	MOV R7,@RW3	MOV R7, @RW3+d16
+C	MOV R0, @RW0+	MOV R0, @RW0+RW7	MOV R1, @RW0+	MOV R1, @RW0+RW7	MOV R2, @RW0+	MOV R2, @RW0+RW7	MOV R3, @RW0+	MOV R3, @RW0+RW7	MOV R4, @RW0+	MOV R4, @RW0+RW7	MOV R5, @RW0+	MOV R5, @RW0+RW7	MOV R6, @RW0+	MOV R6, @RW0+RW7	MOV R7, @RW0+	MOV R7, @RW0+RW7
+D	MOV R0, @RW1+	MOV R0, @RW1+RW7	MOV R1, @RW1+	MOV R1, @RW1+RW7	MOV R2, @RW1+	MOV R2, @RW1+RW7	MOV R3, @RW1+	MOV R3, @RW1+RW7	MOV R4, @RW1+	MOV R4, @RW1+RW7	MOV R5, @RW1+	MOV R5, @RW1+RW7	MOV R6, @RW1+	MOV R6, @RW1+RW7	MOV R7, @RW1+	MOV R7, @RW1+RW7
+E	MOV R0, @RW2+	MOV R0, @PC+d16	MOV R1, @RW2+	MOV R1, @PC+d16	MOV R2, @RW2+	MOV R2, @PC+d16	MOV R3, @RW2+	MOV R3, @PC+d16	MOV R4, @RW2+	MOV R4, @PC+d16	MOV R5, @RW2+	MOV R5, @PC+d16	MOV R6, @RW2+	MOV R6, @PC+d16	MOV R7, @RW2+	MOV R7, @PC+d16
+F	MOV R0, @RW3+	MOV R0, addr16	MOV R1, @RW3+	MOV R1, addr16	MOV R2, @RW3+	MOV R2, addr16	MOV R3, @RW3+	MOV R3, addr16	MOV R4, @RW3+	MOV R4, addr16	MOV R5, @RW3+	MOV R5, addr16	MOV R6, @RW3+	MOV R6, addr16	MOV R7, @RW3+	MOV R7, addr16

Table B.9-17 MOVW RWi, ea Instruction (first byte = $7B_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW RW0 ,@RW0+d8	MOVW RW1,RW0	MOVW RW1 ,@RW0+d8	MOVW RW2,RW0	MOVW RW2 ,@RW0+d8	MOVW RW3,RW0	MOVW RW3 ,@RW0+d8	MOVW RW4,RW0	MOVW RW4 ,@RW0+d8	MOVW RW5,RW0	MOVW RW5 ,@RW0+d8	MOVW RW6,RW0	MOVW RW6 ,@RW0+d8	MOVW RW7,RW0	MOVW RW7 ,@RW0+d8
+ 1	MOVW RW0,RW1	MOVW RW0 ,@RW1+d8	MOVW RW1,RW1	MOVW RW1 ,@RW1+d8	MOVW RW2,RW1	MOVW RW2 ,@RW1+d8	MOVW RW3,RW1	MOVW RW3 ,@RW1+d8	MOVW RW4,RW1	MOVW RW4 ,@RW1+d8	MOVW RW5,RW1	MOVW RW5 ,@RW1+d8	MOVW RW6,RW1	MOVW RW6 ,@RW1+d8	MOVW RW7,RW1	MOVW RW7 ,@RW1+d8
+ 2	MOVW RW0,RW2	MOVW RW0 ,@RW2+d8	MOVW RW1,RW2	MOVW RW1 ,@RW2+d8	MOVW RW2,RW2	MOVW RW2 ,@RW2+d8	MOVW RW3,RW2	MOVW RW3 ,@RW2+d8	MOVW RW4,RW2	MOVW RW4 ,@RW2+d8	MOVW RW5,RW2	MOVW RW5 ,@RW2+d8	MOVW RW6,RW2	MOVW RW6 ,@RW2+d8	MOVW RW7,RW2	MOVW RW7 , @RW2+d8
+ 3	MOVW RW0,RW3	MOVW RW0 ,@RW3+d8	MOVW RW1,RW3	MOVW RW1 ,@RW3+d8	MOVW RW2,RW3	MOVW RW2 ,@RW3+d8	MOVW RW3,RW3	MOVW RW3 ,@RW3+d8	MOVW RW4,RW3	MOVW RW4 ,@RW3+d8	MOVW RW5,RW3	MOVW RW5 ,@RW3+d8	MOVW RW6,RW3	MOVW RW6 ,@RW3+d8	MOVW RW7,RW3	MOVW RW7 ,@RW3+d8
+ 4	MOVW RW0,RW4	MOVW RW0 ,@RW4+d8	MOVW RW1,RW4	MOVW RW1 ,@RW4+d8	MOVW RW2,RW4	MOVW RW2 ,@RW4+d8	MOVW RW3,RW4	MOVW RW3 ,@RW4+d8	MOVW RW4,RW4	MOVW RW4 ,@RW4+d8	MOVW RW5,RW4	MOVW RW5 ,@RW4+d8	MOVW RW6,RW4	MOVW RW6 ,@ RW4+d8	MOVW RW7,RW4	MOVW RW7 ,@RW4+d8
+ 5	MOVW RW0,RW5	MOVW RW0 ,@RW5+d8	MOVW RW1,RW5	MOVW RW1 ,@RW5+d8	MOVW RW2,RW5	MOVW RW2 ,@RW5+d8	MOVW RW3,RW5	MOVW RW3 ,@RW5+d8	MOVW RW4,RW5	MOVW RW4 ,@RW5+d8	MOVW RW5,RW5	MOVW RW5 ,@RW5+d8	MOVW RW6,RW5	MOVW RW6 ,@RW5+d8	MOVW RW7,RW5	MOVW RW7 ,@RW5+d8
+ 6	MOVW RW0,RW6	MOVW RW0 ,@RW6+d8	MOVW RW1,RW6	MOVW RW1 ,@RW6+d8	MOVW RW2,RW6	MOVW RW2 ,@RW6+d8	MOVW RW3,RW6	MOVW RW3 ,@RW6+d8	MOVW RW4,RW6	MOVW RW4 ,@RW6+d8	MOVW RW5,RW6	MOVW RW5 ,@RW6+d8	MOVW RW6,RW6	MOVW RW6 ,@RW6+d8	MOVW RW7,RW6	MOVW RW7 ,@RW6+d8
+ 7	MOVW RW0,RW7	MOVW RW0 ,@RW7+d8	MOVW RW1,RW7	MOVW RW1 ,@RW7+d8	MOVW RW2,RW7	MOVW RW2 ,@RW7+d8	MOVW RW3,RW7	MOVW RW3 ,@RW7+d8	MOVW RW4,RW7	MOVW RW4 ,@RW7+d8	MOVW RW5,RW7	MOVW RW5 ,@RW7+d8	MOVW RW6,RW7	MOVW RW6 ,@RW7+d8	MOVW RW7,RW7	MOVW RW7 ,@RW7+d8
+ 8	MOVW RWO,@RW0	MOVW RWO, @RW0+d16	MOVW RW1,@RW0	MOVW RW1, @RW0+d16	MOVW RW2,@RW0	MOVW RW2, @RW0+d16	MOVW RW3,@RW0	MOVW RW3, @RW0+d16	MOVW RW4,@RW0	MOVW RW4, @RW0+d16	MOVW RW5,@RW0	MOVW RW5, @RW0+d16	MOVW RW6,@RW0	MOVW RW6, @RW0+d16	MOVW RW7,@RW0	MOVW RW7, @RW0+d16
+ 9	MOVW RWO,@RW1	MOVW RWO, @RW1+d16	MOVW RW1,@RW1	MOVW RW1, @RW1+d16	MOVW RW2,@RW1	MOVW RW2, @RW1+d16	MOVW RW3,@RW1	MOVW RW3, @RW1+d16	MOVW RW4,@RW1	MOVW RW4, @RW1+d16	MOVW RW5,@RW1	MOVW RW5, @RW1+d16	MOVW RW6,@RW1	MOVW RW6, @RW1+d16	MOVW RW7,@RW1	MOVW RW7, @RW1+d16
+ A	MOVW RWO,@RW2	MOVW RWO, @RW2+d16	MOVW RW1,@RW2	MOVW RW1, @RW2+d16	MOVW RW2,@RW2	MOVW RW2, @RW2+d16	MOVW RW3,@RW2	MOVW RW3, @RW2+d16	MOVW RW4,@RW2	MOVW RW4, @RW2+d16	MOVW RW5,@RW2	MOVW RW5, @RW2+d16	MOVW RW6,@RW2	MOVW RW6, @RW2+d16	MOVW RW7,@RW2	MOVW RW7, @RW2+d16
+ B	MOVW RWO,@RW3	MOVW RWO, @RW3+d16	MOVW RW1,@RW3	MOVW RW1, @RW3+d16	MOVW RW2,@RW3	MOVW RW2, @RW3+d16	MOVW RW3,@RW3	MOVW RW3, @RW3+d16	MOVW RW4,@RW3	MOVW RW4, @RW3+d16	MOVW RW5,@RW3	MOVW RW5, @RW3+d16	MOVW RW6,@RW3	MOVW RW6, @RW3+d16	MOVW RW7,@RW3	MOVW RW7, @RW3+d16
+ C	MOVW R W0,@RW0+	MOVW RWO, @RW0+RW7	MOVW R W1,@RW0+	MOVW RW1, @RW0+RW7	MOVW R W2,@RW0+	MOVW RW2, @RW0+RW7	MOVW R W3,@RW0+	MOVW RW3, @RW0+RW7	MOVW R W4,@RW0+	MOVW RW4, @RW0+RW7	MOVW R W5,@RW0+	MOVW RW5, @RW0+RW7	MOVW R W6,@RW0+	MOVW Å@R @RW0+RW7	MOVW R W7,@RW0+	MOVW Å@R @RW0+RW7
+ D	MOVW R W0,@RW1+	MOVW RWO, @RW1+RW7	MOVW R W1,@RW1+	MOVW RW1, @RW1+RW7	MOVW R W2,@RW1+	MOVW RW2, @RW1+RW7	MOVW R W3,@RW1+	MOVW RW3, @RW1+RW7	MOVW R W4,@RW1+	MOVW RW4, @RW1+RW7	MOVW R W5,@RW1+	MOVW RW5, @RW1+RW7	MOVW R W6,@RW1+	MOVW RW6, @RW1+RW7	MOVW R W7,@RW1+	MOVW RW7, @RW1+RW7
+ E	MOVW R W0,@RW2+	MOVW RW0 ,@PC+d16	MOVW R W1,@RW2+	MOVW RW1 ,@PC+d16	MOVW R W2,@RW2+	MOVW RW2 ,@PC+d16	MOVW R W3,@RW2+	MOVW RW3 ,@PC+d16	MOVW R W4,@RW2+	MOVW RW4 ,@PC+d16	MOVW R W5,@RW2+	MOVW RW5 ,@PC+d16	MOVW R W6,@RW2+	MOVW RW6 ,@PC+d16	MOVW R W7,@RW2+	MOVW RW7 ,@PC+d16
+ F	MOVW R W0,@RW3+	MOVW RW 0,addr16	MOVW R W1,@RW3+	MOVW RW 1,addr16	MOVW R W2,@RW3+	MOVW RW 2,addr16	MOVW R W3,@RW3+	MOVW RW 3,addr16	MOVW R W4,@RW3+	MOVW RW 4,addr16	MOVW R W5,@RW3+	MOVW RW 5,addr16	MOVW R W6,@RW3+	MOVW RW 6,addr16	MOVW R W7,@RW3+	MOVW RW 7,addr16

Table B.9-18 MOV Ri, ea Instruction (first byte = 7C$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0,R0	MOV @R W0+d8,R0	MOV R0,R1	MOV @R W0+d8,R1	MOV R0,R2	MOV @R W0+d8,R2	MOV R0,R3	MOV @R W0+d8,R3	MOV R0,R4	MOV @R W0+d8,R4	MOV R0,R5	MOV @R W0+d8,R5	MOV R0,R6	MOV @R W0+d8,R6	MOV R0,R7	MOV @R W0+d8,R7
+ 1	MOV R1,R0	MOV @R W1+d8,R0	MOV R1,R1	MOV @R W1+d8,R1	MOV R1,R2	MOV @R W1+d8,R2	MOV R1,R3	MOV @R W1+d8,R3	MOV R1,R4	MOV @R W1+d8,R4	MOV R1,R5	MOV @R W1+d8,R5	MOV R1,R6	MOV @R W1+d8,R6	MOV R1,R7	MOV @R W1+d8,R7
+ 2	MOV R2,R0	MOV @R W2+d8,R0	MOV R2,R1	MOV @R W2+d8,R1	MOV R2,R2	MOV @R W2+d8,R2	MOV R2,R3	MOV @R W2+d8,R3	MOV R2,R4	MOV @R W2+d8,R4	MOV R2,R5	MOV @R W2+d8,R5	MOV R2,R6	MOV @R W2+d8,R6	MOV R2,R7	MOV @R W2+d8,R7
+ 3	MOV R3,R0	MOV @R W3+d8,R0	MOV R3,R1	MOV @R W3+d8,R1	MOV R3,R2	MOV @R W3+d8,R2	MOV R3,R3	MOV @R W3+d8,R3	MOV R3,R4	MOV @R W3+d8,R4	MOV R3,R5	MOV @R W3+d8,R5	MOV R3,R6	MOV @R W3+d8,R6	MOV R3,R7	MOV @R W3+d8,R7
+ 4	MOV R4,R0	MOV @R W4+d8,R0	MOV R4,R1	MOV @R W4+d8,R1	MOV R4,R2	MOV @R W4+d8,R2	MOV R4,R3	MOV @R W4+d8,R3	MOV R4,R4	MOV @R W4+d8,R4	MOV R4,R5	MOV @R W4+d8,R5	MOV R4,R6	MOV @R W4+d8,R6	MOV R4,R7	MOV @R W4+d8,R7
+ 5	MOV R5,R0	MOV @R W5+d8,R0	MOV R5,R1	MOV @R W5+d8,R1	MOV R5,R2	MOV @R W5+d8,R2	MOV R5,R3	MOV @R W5+d8,R3	MOV R5,R4	MOV @R W5+d8,R4	MOV R5,R5	MOV @R W5+d8,R5	MOV R5,R6	MOV @R W5+d8,R6	MOV R5,R7	MOV @R W5+d8,R7
+ 6	MOV R6,R0	MOV @R W6+d8,R0	MOV R6,R1	MOV @R W6+d8,R1	MOV R6,R2	MOV @R W6+d8,R2	MOV R6,R3	MOV @R W6+d8,R3	MOV R6,R4	MOV @R W6+d8,R4	MOV R6,R5	MOV @R W6+d8,R5	MOV R6,R6	MOV @R W6+d8,R6	MOV R6,R7	MOV @R W6+d8,R7
+ 7	MOV R7,R0	MOV @R W7+d8,R0	MOV R7,R1	MOV @R W7+d8,R1	MOV R7,R2	MOV @R W7+d8,R2	MOV R7,R3	MOV @R W7+d8,R3	MOV R7,R4	MOV @R W7+d8,R4	MOV R7,R5	MOV @R W7+d8,R5	MOV R7,R6	MOV @R W7+d8,R6	MOV R7,R7	MOV @R W7+d8,R7
+ 8	MOV @RW0,R0	MOV @RW 0+d16,R0	MOV @RW0,R1	MOV @RW 0+d16,R1	MOV @RW0,R2	MOV @RW 0+d16,R2	MOV @RW0,R3	MOV @RW 0+d16,R3	MOV @RW0,R4	MOV @RW 0+d16,R4	MOV @RW0,R5	MOV @RW 0+d16,R5	MOV @RW0,R6	MOV @RW 0+d16,R6	MOV @RW0,R7	MOV @RW 0+d16,R7
+ 9	MOV @RW1,R0	MOV @RW 1+d16,R0	MOV @RW1,R1	MOV @RW 1+d16,R1	MOV @RW1,R2	MOV @RW 1+d16,R2	MOV @RW1,R3	MOV @RW 1+d16,R3	MOV @RW1,R4	MOV @RW 1+d16,R4	MOV @RW1,R5	MOV @RW 1+d16,R5	MOV @RW1,R6	MOV @RW 1+d16,R6	MOV @RW1,R7	MOV @RW 1+d16,R7
+ A	MOV @RW2,R0	MOV @RW 2+d16,R0	MOV @RW2,R1	MOV @RW 2+d16,R1	MOV @RW2,R2	MOV @RW 2+d16,R2	MOV @RW2,R3	MOV @RW 2+d16,R3	MOV @RW2,R4	MOV @RW 2+d16,R4	MOV @RW2,R5	MOV @RW 2+d16,R5	MOV @RW2,R6	MOV @RW 2+d16,R6	MOV @RW2,R7	MOV @RW 2+d16,R7
+ B	MOV @RW3,R0	MOV @RW 3+d16,R0	MOV @RW3,R1	MOV @RW 3+d16,R1	MOV @RW3,R2	MOV @RW 3+d16,R2	MOV @RW3,R3	MOV @RW 3+d16,R3	MOV @RW3,R4	MOV @RW 3+d16,R4	MOV @RW3,R5	MOV @RW 3+d16,R5	MOV @RW3,R6	MOV @RW 3+d16,R6	MOV @RW3,R7	MOV @RW 3+d16,R7
+ C	MOV @RW0+,R0	MOV @RW 0+RW7,R0	MOV @RW0+,R1	MOV @RW 0+RW7,R1	MOV @RW0+,R2	MOV @RW 0+RW7,R2	MOV @RW0+,R3	MOV @RW 0+RW7,R3	MOV @RW0+,R4	MOV @RW 0+RW7,R4	MOV @RW0+,R5	MOV @RW 0+RW7,R5	MOV @RW0+,R6	MOV @RW 0+RW7,R6	MOV @RW0+,R7	MOV @RW 0+RW7,R7
+ D	MOV @RW1+,R0	MOV @RW 1+RW7,R0	MOV @RW1+,R1	MOV @RW 1+RW7,R1	MOV @RW1+,R2	MOV @RW 1+RW7,R2	MOV @RW1+,R3	MOV @RW 1+RW7,R3	MOV @RW1+,R4	MOV @RW 1+RW7,R4	MOV @RW1+,R5	MOV @RW 1+RW7,R5	MOV @RW1+,R6	MOV @RW 1+RW7,R6	MOV @RW1+,R7	MOV @RW 1+RW7,R7
+ E	MOV @RW2+,R0	MOV P C+d16,R0	MOV @RW2+,R1	MOV P C+d16,R1	MOV @RW2+,R2	MOV P C+d16,R2	MOV @RW2+,R3	MOV P C+d16,R3	MOV @RW2+,R4	MOV P C+d16,R4	MOV @RW2+,R5	MOV P C+d16,R5	MOV @RW2+,R6	MOV P C+d16,R6	MOV @RW2+,R7	MOV P C+d16,R7
+ F	MOV @RW3+,R0	MOV addr16,R0	MOV @RW3+,R1	MOV addr16,R1	MOV @RW3+,R2	MOV addr16,R2	MOV @RW3+,R3	MOV addr16,R3	MOV @RW3+,R4	MOV addr16,R4	MOV @RW3+,R5	MOV addr16,R5	MOV @RW3+,R6	MOV addr16,R6	MOV @RW3+,R7	MOV addr16,R7

Table B.9-19 MOVW ea, Rwi Instruction (first byte = $7D_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW @RW 0+d8,RW0	MOVW RW0,RW1	MOVW @RW 0+d8,RW1	MOVW RW0,RW2	MOVW @RW 0+d8,RW2	MOVW RW0,RW3	MOVW @RW 0+d8,RW3	MOVW RW0,RW4	MOVW @RW 0+d8,RW4	MOVW RW0,RW5	MOVW @RW 0+d8,RW5	MOVW RW0,RW6	MOVW @RW 0+d8,RW6	MOVW RW0,RW7	MOVW @RW 0+d8,RW7
+ 1	MOVW RW1,RW0	MOVW @RW 1+d8,RW0	MOVW RW1,RW1	MOVW @RW 1+d8,RW1	MOVW RW1,RW2	MOVW @RW 1+d8,RW2	MOVW RW1,RW3	MOVW @RW 1+d8,RW3	MOVW RW1,RW4	MOVW @RW 1+d8,RW4	MOVW RW1,RW5	MOVW @RW 1+d8,RW5	MOVW RW1,RW6	MOVW @RW 1+d8,RW6	MOVW RW1,RW7	MOVW @RW 1+d8,RW7
+ 2	MOVW RW2,RW0	MOVW @RW 2+d8,RW0	MOVW RW2,RW1	MOVW @RW 2+d8,RW1	MOVW RW2,RW2	MOVW @RW 2+d8,RW2	MOVW RW2,RW3	MOVW @RW 2+d8,RW3	MOVW RW2,RW4	MOVW @RW 2+d8,RW4	MOVW RW2,RW5	MOVW @RW 2+d8,RW5	MOVW RW2,RW6	MOVW @RW 2+d8,RW6	MOVW RW2,RW7	MOVW @RW 2+d8,RW7
+ 3	MOVW RW3,RW0	MOVW @RW 3+d8,RW0	MOVW RW3,RW1	MOVW @RW 3+d8,RW1	MOVW RW3,RW2	MOVW @RW 3+d8,RW2	MOVW RW3,RW3	MOVW @RW 3+d8,RW3	MOVW RW3,RW4	MOVW @RW 3+d8,RW4	MOVW RW3,RW5	MOVW @RW 3+d8,RW5	MOVW RW3,RW6	MOVW @RW 3+d8,RW6	MOVW RW3,RW7	MOVW @RW 3+d8,RW7
+ 4	MOVW RW4,RW0	MOVW @RW 4+d8,RW0	MOVW RW4,RW1	MOVW @RW 4+d8,RW1	MOVW RW4,RW2	MOVW @RW 4+d8,RW2	MOVW RW4,RW3	MOVW @RW 4+d8,RW3	MOVW RW4,RW4	MOVW @RW 4+d8,RW4	MOVW RW4,RW5	MOVW @RW 4+d8,RW5	MOVW RW4,RW6	MOVW @RW 4+d8,RW6	MOVW RW4,RW7	MOVW @RW 4+d8,RW7
+ 5	MOVW RW5,RW0	MOVW @RW 5+d8,RW0	MOVW RW5,RW1	MOVW @RW 5+d8,RW1	MOVW RW5,RW2	MOVW @RW 5+d8,RW2	MOVW RW5,RW3	MOVW @RW 5+d8,RW3	MOVW RW5,RW4	MOVW @RW 5+d8,RW4	MOVW RW5,RW5	MOVW @RW 5+d8,RW5	MOVW RW5,RW6	MOVW @RW 5+d8,RW6	MOVW RW5,RW7	MOVW @RW 5+d8,RW7
+ 6	MOVW RW6,RW0	MOVW @RW 6+d8,RW0	MOVW RW6,RW1	MOVW @RW 6+d8,RW1	MOVW RW6,RW2	MOVW @RW 6+d8,RW2	MOVW RW6,RW3	MOVW @RW 6+d8,RW3	MOVW RW6,RW4	MOVW @RW 6+d8,RW4	MOVW RW6,RW5	MOVW @RW 6+d8,RW5	MOVW RW6,RW6	MOVW @RW 6+d8,RW6	MOVW RW6,RW7	MOVW @RW 6+d8,RW7
+ 7	MOVW RW7,RW0	MOVW @RW 7+d8,RW0	MOVW RW7,RW1	MOVW @RW 7+d8,RW1	MOVW RW7,RW2	MOVW @RW 7+d8,RW2	MOVW RW7,RW3	MOVW @RW 7+d8,RW3	WOVW RW7,RW4	MOVW @RW 7+d8,RW4	WOVW RW7,RW5	MOVW @RW 7+d8,RW5	WOVW RW7,RW6	MOVW @RW 7+d8,RW6	WOVW RW7,RW7	MOVW @RW 7+d8,RW7
+ 8	MOVW @RW0,RW0	MOVW @RW0 +d16,RW0	MOVW @RW0,RW1	MOVW @RW0 +d16,RW1	MOVW @RW0,RW2	MOVW @RW0 +d16,RW2	MOVW @RW0,RW3	MOVW @RW0 +d16,RW3	MOVW @RW0,RW4	MOVW @RW0 +d16,RW4	MOVW @RW0,RW5	MOVW @RW0 +d16,RW5	MOVW @RW0,RW6	MOVW @RW0 +d16,RW6	MOVW @RW0,RW7	MOVW @RW0 +d16,RW7
+ 9	MOVW @RW1,RW0	MOVW @RW1 +d16,RW0	MOVW @RW1,RW1	MOVW @RW1 +d16,RW1	MOVW @RW1,RW2	MOVW @RW1 +d16,RW2	MOVW @RW1,RW3	MOVW @RW1 +d16,RW3	MOVW @RW1,RW4	MOVW @RW1 +d16,RW4	MOVW @RW1,RW5	MOVW @RW1 +d16,RW5	MOVW @RW1,RW6	MOVW @RW1 +d16,RW6	MOVW @RW1,RW7	MOVW @RW1 +d16,RW7
+ A	MOVW @RW2,RW0	MOVW @RW2 +d16,RW0	MOVW @RW2,RW1	MOVW @RW2 +d16,RW1	MOVW @RW2,RW2	MOVW @RW2 +d16,RW2	MOVW @RW2,RW3	MOVW @RW2 +d16,RW3	MOVW @RW2,RW4	MOVW @RW2 +d16,RW4	MOVW @RW2,RW5	MOVW @RW2 +d16,RW5	MOVW @RW2,RW6	MOVW @RW2 +d16,RW6	MOVW @RW2,RW7	MOVW @RW2 +d16,RW7
+ B	MOVW @RW3,RW0	MOVW @RW3 +d16,RW0	MOVW @RW3,RW1	MOVW @RW3 +d16,RW1	MOVW @RW3,RW2	MOVW @RW3 +d16,RW2	MOVW @RW3,RW3	MOVW @RW3 +d16,RW3	MOVW @RW3,RW4	MOVW @RW3 +d16,RW4	MOVW @RW3,RW5	MOVW @RW3 +d16,RW5	MOVW @RW3,RW6	MOVW @RW3 +d16,RW6	MOVW @RW3,RW7	MOVW @RW3 +d16,RW7
+ C	MOVW @ RW0+,RW0	MOVW @RW0 +RW7,RW0	MOVW @ RW0+,RW1	MOVW @RW0 +RW7,RW1	MOVW @ RW0+,RW2	MOVW @RW0 +RW7,RW2	MOVW @ RW0+,RW3	MOVW @RW0 +RW7,RW3	MOVW @ RW0+,RW4	MOVW @RW0 +RW7,RW4	MOVW @ RW0+,RW5	MOVW @RW0 +RW7,RW5	MOVW @ RW0+,RW6	MOVW @RW0 +RW7,RW6	MOVW @ RW0+,RW7	MOVW @RW0 +RW7,RW7
+ D	MOVW @ RW1+,RW0	MOVW @RW1 +RW7,RW0	MOVW @ RW1+,RW1	MOVW @RW1 +RW7,RW1	MOVW @ RW1+,RW2	MOVW @RW1 +RW7,RW2	MOVW @ RW1+,RW3	MOVW @RW1 +RW7,RW3	MOVW @ RW1+,RW4	MOVW @RW1 +RW7,RW4	MOVW @ RW1+,RW5	MOVW @RW1 +RW7,RW5	MOVW @ RW1+,RW6	MOVW @RW1 +RW7,RW6	MOVW @ RW1+,RW7	MOVW @RW1 +RW7,RW7
+ E	MOVW @ RW2+,RW0	MOVW @PC+ d16,RW0	MOVW @ RW2+,RW1	MOVW @PC+ d16,RW1	MOVW @ RW2+,RW2	MOVW @PC+ d16,RW2	MOVW @ RW2+,RW3	MOVW @PC+ d16,RW3	MOVW @ RW2+,RW4	MOVW @PC+ d16,RW4	MOVW @ RW2+,RW5	MOVW @PC+ d16,RW5	MOVW @ RW2+,RW6	MOVW @PC+ d16,RW6	MOVW @ RW2+,RW7	MOVW @PC+ d16,RW7
+ F	MOVW @ RW3+,RW0	MOVW addr16,RW0	MOVW @ RW3+,RW1	MOVW addr16,RW1	MOVW @ RW3+,RW2	MOVW addr16,RW2	MOVW @ RW3+,RW3	MOVW addr16,RW3	MOVW @ RW3+,RW4	MOVW addr16,RW4	MOVW @ RW3+,RW5	MOVW addr16,RW5	MOVW @ RW3+,RW6	MOVW addr16,RW6	MOVW @ RW3+,RW7	MOVW addr16,RW7

Table B.9-20 XCH Ri, ea Instruction (first byte = 7E$_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCH R0,R0	XCH R0, @RW0+d8	XCH R1,R0	XCH R1, @RW0+d8	XCH R2,R0	XCH R2, @RW0+d8	XCH R3,R0	XCH R3, @RW0+d8	XCH R4,R0	XCH R4, @RW0+d8	XCH R5,R0	XCH R5, @RW0+d8	XCH R6,R0	XCH R6, @RW0+d8	XCH R7,R0	XCH R7, @RW0+d8
+ 1	XCH R0,R1	XCH R0, @RW1+d8	XCH R1,R1	XCH R1, @RW1+d8	XCH R2,R1	XCH R2, @RW1+d8	XCH R3,R1	XCH R3, @RW1+d8	XCH R4,R1	XCH R4, @RW1+d8	XCH R5,R1	XCH R5, @RW1+d8	XCH R6,R1	XCH R6, @RW1+d8	XCH R7,R1	NOTW R7, @RW1+d8
+ 2	XCH R0,R2	XCH R0, @RW2+d8	XCH R1,R2	XCH R1, @RW2+d8	XCH R2,R2	XCH R2, @RW2+d8	XCH R3,R2	XCH R3, @RW2+d8	XCH R4,R2	XCH R4, @RW2+d8	XCH R5,R2	XCH R5, @RW2+d8	XCH R6,R2	XCH R6, @RW2+d8	XCH R7,R2	XCH R7, @RW2+d8
+ 3	XCH R0,R3	XCH R0, @RW3+d8	XCH R1,R3	XCH R1, @RW3+d8	XCH R2,R3	XCH R2, @RW3+d8	XCH R3,R3	XCH R3, @RW3+d8	XCH R4,R3	XCH R4, @RW3+d8	XCH R5,R3	XCH R5, @RW3+d8	XCH R6,R3	XCH R6, @RW3+d8	XCH R7,R3	NOTW R7, @RW3+d8
+ 4	XCH R0,R4	XCH R0, @RW4+d8	XCH R1,R4	XCH R1, @RW4+d8	XCH R2,R4	XCH R2, @RW4+d8	XCH R3,R4	XCH R3, @RW4+d8	XCH R4,R4	XCH R4, @RW4+d8	XCH R5,R4	XCH R5, @RW4+d8	XCH R6,R4	XCH R6, @RW4+d8	XCH R7,R4	NOTW R7, @RW4+d8
+ 5	XCH R0,R5	XCH R0, @RW5+d8	XCH R1,R5	XCH R1, @RW5+d8	XCH R2,R5	XCH R2, @RW5+d8	XCH R3,R5	XCH R3, @RW5+d8	XCH R4,R5	XCH R4, @RW5+d8	XCH R5,R5	XCH R5, @RW5+d8	XCH R6,R5	XCH R6, @RW5+d8	XCH R7,R5	XCH R7, @RW5+d8
+ 6	XCH R0,R6	XCH R0, @RW6+d8	XCH R1,R6	XCH R1, @RW6+d8	XCH R2,R6	XCH R2, @RW6+d8	XCH R3,R6	XCH R3, @RW6+d8	XCH R4,R6	XCH R4, @RW6+d8	XCH R5,R6	XCH R5, @RW6+d8	XCH R6,R6	XCH R6, @RW6+d8	XCH R7,R6	XCH R7, @RW6+d8
+ 7	XCH R0,R7	XCH R0, @RW7+d8	XCH R1,R7	XCH R1, @RW7+d8	XCH R2,R7	XCH R2, @RW7+d8	XCH R3,R7	XCH R3, @RW7+d8	XCH R4,R7	XCH R4, @RW7+d8	XCH R5,R7	XCH R5, @RW7+d8	XCH R6,R7	XCH R6, @RW7+d8	XCH R7,R7	XCH R7, @RW7+d8
+ 8	XCH R0,@RW0	XCH R0, @RW0+d16	XCH R1,@RW0	XCH R1, @RW0+d16	XCH R2,@RW0	XCH R2, @RW0+d16	XCH R3,@RW0	XCH R3, @RW0+d16	XCH R4,@RW0	XCH R4, @RW0+d16	XCH R5,@RW0	XCH R5, @RW0+d16	XCH R6,@RW0	XCH R6, @RW0+d16	XCH R7,@RW0	XCH R7, @RW0+d16
+ 9	XCH R0,@RW1	XCH R0, @RW1+d16	XCH R1,@RW1	XCH R1, @RW1+d16	XCH R2,@RW1	XCH R2, @RW1+d16	XCH R3,@RW1	XCH R3, @RW1+d16	XCH R4,@RW1	XCH R4, @RW1+d16	XCH R5,@RW1	XCH R5, @RW1+d16	XCH R6,@RW1	XCH R6, @RW1+d16	XCH R7,@RW1	XCH R7, @RW1+d16
+ A	XCH R0,@RW2	XCH R0, W2+d16,A	XCH R1,@RW2	XCH R1, W2+d16,A	XCH R2,@RW2	XCH R2, W2+d16,A	XCH R3,@RW2	XCH R3, W2+d16,A	XCH R4,@RW2	XCH R4, W2+d16,A	XCH R5,@RW2	XCH R5, W2+d16,A	XCH R6,@RW2	XCH R6, W2+d16,A	XCH R7,@RW2	XCH R7, W2+d16,A
+ B	XCH R0,@RW3	XCH R0, @RW3+d16	XCH R1,@RW3	XCH R1, @RW3+d16	XCH R2,@RW3	XCH R2, @RW3+d16	XCH R3,@RW3	XCH R3, @RW3+d16	XCH R4,@RW3	XCH R4, @RW3+d16	XCH R5,@RW3	XCH R5, @RW3+d16	XCH R6,@RW3	XCH R6, @RW3+d16	XCH R7,@RW3	XCH R7, @RW3+d16
+ C	XCH R0,@RW0+	XCH R0, @RW0+RW7	XCH R1,@RW0+	XCH R1, @RW0+RW7	XCH R2,@RW0+	XCH R2, @RW0+RW7	XCH R3,@RW0+	XCH R3, @RW0+RW7	XCH R4,@RW0+	XCH R4, @RW0+RW7	XCH R5,@RW0+	XCH R5, @RW0+RW7	XCH R6,@RW0+	XCH R6, @RW0+RW7	XCH R7,@RW0+	XCH R7, @RW0+RW7
+ D	XCH R0,@RW1+	XCH R0, @RW1+RW7	XCH R1,@RW1+	XCH R1, @RW1+RW7	XCH R2,@RW1+	XCH R2, @RW1+RW7	XCH R3,@RW1+	XCH R3, @RW1+RW7	XCH R4,@RW1+	XCH R4, @RW1+RW7	XCH R5,@RW1+	XCH R5, @RW1+RW7	XCH R6,@RW1+	XCH R6, @RW1+RW7	XCH R7,@RW1+	XCH R7, @RW1+RW7
+ E	XCH R0,@RW2+	XCH R0, @PC+d16	XCH R1,@RW2+	XCH R1, @PC+d16	XCH R2,@RW2+	XCH R2, @PC+d16	XCH R3,@RW2+	XCH R3, @PC+d16	XCH R4,@RW2+	XCH R4, @PC+d16	XCH R5,@RW2+	XCH R5, @PC+d16	XCH R6,@RW2+	XCH R6, @PC+d16	XCH R7,@RW2+	XCH R7, @PC+d16
+ F	XCH R0,@RW3+	XCH R0, addr16	XCH R1,@RW3+	XCH R1, addr16	XCH R2,@RW3+	XCH R2, addr16	XCH R3,@RW3+	XCH R3, addr16	XCH R4,@RW3+	XCH R4, addr16	XCH R5,@RW3+	XCH R5, addr16	XCH R6,@RW3+	XCH R6, addr16	XCH R7,@RW3+	XCH R7, addr16

Table B.9-21 XCHW RWi, ea Instruction (first byte = 7F$_H$)

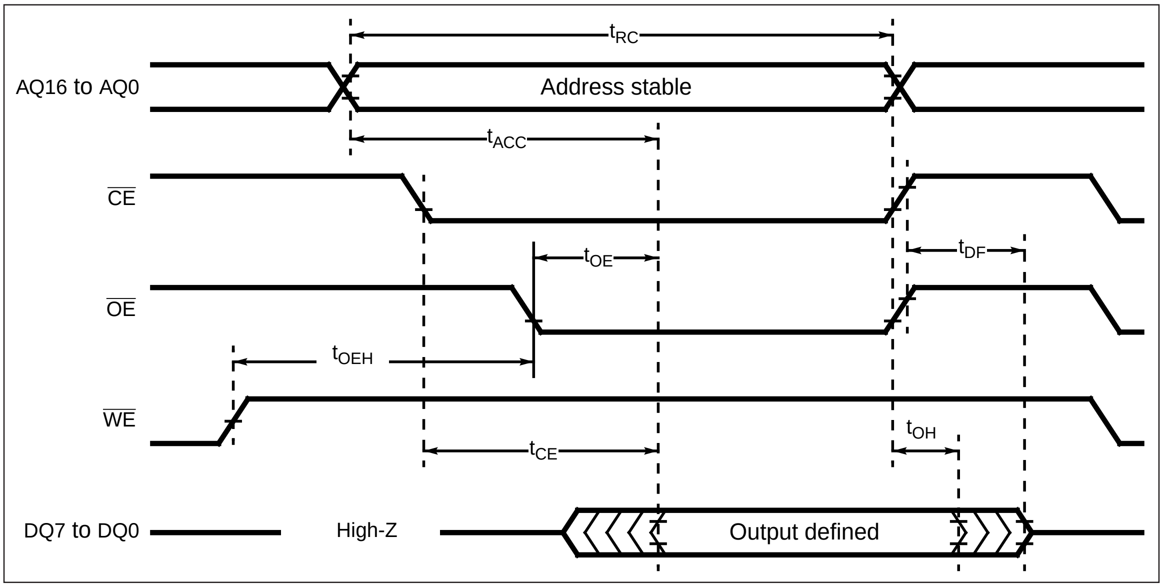
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCHW RW0,RW0	XCHW RW0, @RW0+d8	XCHW RW1,RW0	XCHW RW1, @RW0+d8	XCHW RW2,RW0	XCHW RW2, @RW0+d8	XCHW RW3,RW0	XCHW RW3, @RW0+d8	XCHW RW4,RW0	XCHW RW4, @RW0+d8	XCHW RW5,RW0	XCHW RW5, @RW0+d8	XCHW RW6,RW0	XCHW RW6, @RW0+d8	XCHW RW7,RW0	XCHW RW7, @RW0+d8
+ 1	XCHW RW0,RW1	XCHW RW0, @RW1+d8	XCHW RW1,RW1	XCHW RW1, @RW1+d8	XCHW RW2,RW1	XCHW RW2, @RW1+d8	XCHW RW3,RW1	XCHW RW3, @RW1+d8	XCHW RW4,RW1	XCHW RW4, @RW1+d8	XCHW RW5,RW1	XCHW RW5, @RW1+d8	XCHW RW6,RW1	XCHW RW6, @RW1+d8	XCHW RW7,RW1	XCHW RW7, @RW1+d8
+ 2	XCHW RW0,RW2	XCHW RW0, @RW2+d8	XCHW RW1,RW2	XCHW RW1, @RW2+d8	XCHW RW2,RW2	XCHW RW2, @RW2+d8	XCHW RW3,RW2	XCHW RW3, @RW2+d8	XCHW RW4,RW2	XCHW RW4, @RW2+d8	XCHW RW5,RW2	XCHW RW5, @RW2+d8	XCHW RW6,RW2	XCHW RW6, @RW2+d8	XCHW RW7,RW2	XCHW RW7, @RW2+d8
+ 3	XCHW RW0,RW3	XCHW RW0, @RW3+d8	XCHW RW1,RW3	XCHW RW1, @RW3+d8	XCHW RW2,RW3	XCHW RW2, @RW3+d8	XCHW RW3,RW3	XCHW RW3, @RW3+d8	XCHW RW4,RW3	XCHW RW4, @RW3+d8	XCHW RW5,RW3	XCHW RW5, @RW3+d8	XCHW RW6,RW3	XCHW RW6, @RW3+d8	XCHW RW7,RW3	XCHW RW7, @RW3+d8
+ 4	XCHW RW0,RW4	XCHW RW0, @RW4+d8	XCHW RW1,RW4	XCHW RW1, @RW4+d8	XCHW RW2,RW4	XCHW RW2, @RW4+d8	XCHW RW3,RW4	XCHW RW3, @RW4+d8	XCHW RW4,RW4	XCHW RW4, @RW4+d8	XCHW RW5,RW4	XCHW RW5, @RW4+d8	XCHW RW6,RW4	XCHW RW6, @RW4+d8	XCHW RW7,RW4	XCHW RW7, @RW4+d8
+ 5	XCHW RW0,RW5	XCHW RW0, @RW5+d8	XCHW RW1,RW5	XCHW RW1, @RW5+d8	XCHW RW2,RW5	XCHW RW2, @RW5+d8	XCHW RW3,RW5	XCHW RW3, @RW5+d8	XCHW RW4,RW5	XCHW RW4, @RW5+d8	XCHW RW5,RW5	XCHW RW5, @RW5+d8	XCHW RW6,RW5	XCHW RW6, @RW5+d8	XCHW RW7,RW5	XCHW RW7, @RW5+d8
+ 6	XCHW RW0,RW6	XCHW RW0, @RW6+d8	XCHW RW1,RW6	XCHW RW1, @RW6+d8	XCHW RW2,RW6	XCHW RW2, @RW6+d8	XCHW RW3,RW6	XCHW RW3, @RW6+d8	XCHW RW4,RW6	XCHW RW4, @RW6+d8	XCHW RW5,RW6	XCHW RW5, @RW6+d8	XCHW RW6,RW6	XCHW RW6, @RW6+d8	XCHW RW7,RW6	XCHW RW7, @RW6+d8
+ 7	XCHW RW0,RW7	XCHW RW0, @RW7+d8	XCHW RW1,RW7	XCHW RW1, @RW7+d8	XCHW RW2,RW7	XCHW RW2, @RW7+d8	XCHW RW3,RW7	XCHW RW3, @RW7+d8	XCHW RW4,RW7	XCHW RW4, @RW7+d8	XCHW RW5,RW7	XCHW RW5, @RW7+d8	XCHW RW6,RW7	XCHW RW6, @RW7+d8	XCHW RW7,RW7	XCHW RW7, @RW7+d8
+ 8	XCHW RW0,@RW0	XCHW RW0, @RW0+d16	XCHW RW1,@RW0	XCHW RW1, @RW0+d16	XCHW RW2,@RW0	XCHW RW2, @RW0+d16	XCHW RW3,@RW0	XCHW RW3, @RW0+d16	XCHW RW4,@RW0	XCHW RW4, @RW0+d16	XCHW RW5,@RW0	XCHW RW5, @RW0+d16	XCHW RW6,@RW0	XCHW RW6, @RW0+d16	XCHW RW7,@RW0	XCHW RW7, @RW0+d16
+ 9	XCHW RW0,@RW1	XCHW RW0, @RW1+d16	XCHW RW1,@RW1	XCHW RW1, @RW1+d16	XCHW RW2,@RW1	XCHW RW2, @RW1+d16	XCHW RW3,@RW1	XCHW RW3, @RW1+d16	XCHW RW4,@RW1	XCHW RW4, @RW1+d16	XCHW RW5,@RW1	XCHW RW5, @RW1+d16	XCHW RW6,@RW1	XCHW RW6, @RW1+d16	XCHW RW7,@RW1	XCHW RW7, @RW1+d16
+ A	XCHW RW0,@RW2	XCHW RW0, @RW2+d16	XCHW RW1,@RW2	XCHW RW1, @RW2+d16	XCHW RW2,@RW2	XCHW RW2, @RW2+d16	XCHW RW3,@RW2	XCHW RW3, @RW2+d16	XCHW RW4,@RW2	XCHW RW4, @RW2+d16	XCHW RW5,@RW2	XCHW RW5, @RW2+d16	XCHW RW6,@RW2	XCHW RW6, @RW2+d16	XCHW RW7,@RW2	XCHW RW7, @RW2+d16
+ B	XCHW RW0,@RW3	XCHW RW0, @RW3+d16	XCHW RW1,@RW3	XCHW RW1, @RW3+d16	XCHW RW2,@RW3	XCHW RW2, @RW3+d16	XCHW RW3,@RW3	XCHW RW3, @RW3+d16	XCHW RW4,@RW3	XCHW RW4, @RW3+d16	XCHW RW5,@RW3	XCHW RW5, @RW3+d16	XCHW RW6,@RW3	XCHW RW6, @RW3+d16	XCHW RW7,@RW3	XCHW RW7, @RW3+d16
+ C	XCHW R W0,@RW0+	XCHW RW0, @RW0+RW7	XCHW R W1,@RW0+	XCHW RW1, @RW0+RW7	XCHW R W2,@RW0+	XCHW RW2, @RW0+RW7	XCHW R W3,@RW0+	XCHW RW3, @RW0+RW7	XCHW R W4,@RW0+	XCHW RW4, @RW0+RW7	XCHW R W5,@RW0+	XCHW RW5, @RW0+RW7	XCHW R W6,@RW0+	XCHW RW6, @RW0+RW7	XCHW R W7,@RW0+	XCHW RW7, @RW0+RW7
+ D	XCHW R WO,@RW1+	XCHW RW0, @RW1+RW7	XCHW R W1,@RW1+	XCHW RW1, @RW1+RW7	XCHW R W2,@RW1+	XCHW RW2, @RW1+RW7	XCHW R W3,@RW1+	XCHW RW3, @RW1+RW7	XCHW R W4,@RW1+	XCHW RW4, @RW1+RW7	XCHW R W5,@RW1+	XCHW RW5, @RW1+RW7	XCHW R W6,@RW1+	XCHW RW6, @RW1+RW7	XCHW R W7,@RW1+	XCHW RW7, @RW1+RW7
+ E	XCHW R WO,@RW2+	XCHW RW0, @PC+d16	XCHW R W1,@RW2+	XCHW RW1, @PC+d16	XCHW R W2,@RW2+	XCHW RW2, @PC+d16	XCHW R W3,@RW2+	XCHW RW3, @PC+d16	XCHW R W4,@RW2+	XCHW RW4, @PC+d16	XCHW R W5,@RW2+	XCHW RW5, @PC+d16	XCHW R W6,@RW2+	XCHW RW6, @PC+d16	XCHW R W7,@RW2+	XCHW RW7, @PC+d16
+ F	XCHW R WO,@RW3+	XCHW RW0, addr16	XCHW R W1,@RW3+	XCHW RW1, addr16	XCHW R W2,@RW3+	XCHW RW2, addr16	XCHW R W3,@RW3+	XCHW RW3, addr16	XCHW R W4,@RW3+	XCHW RW4, addr16	XCHW R W5,@RW3+	XCHW RW5, addr16	XCHW R W6,@RW3+	XCHW RW6, addr16	XCHW R W7,@RW3+	XCHW RW7, addr16

APPENDIX C Timing Diagrams in Flash Memory Mode

Each timing diagram for the external pins of the Flash devices in MB90360 series during Flash Memory mode is shown below.

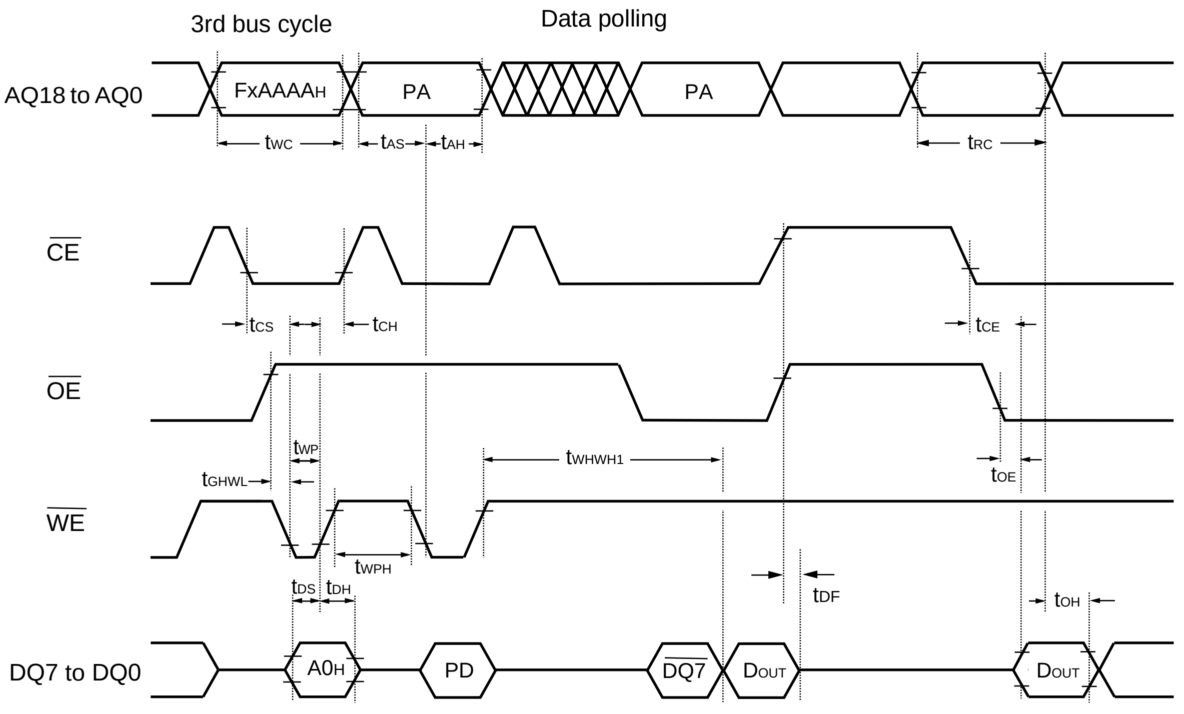
■ Data Read by Read Access

Figure C-1 Timing Diagram for Read Access



■ Write, Data Polling, Read (WE control)

Figure C-2 Write, Data Polling, Read (WE control)



PA : Write address

PD : Write data

$\overline{DQ_7}$: Reverse output of write data

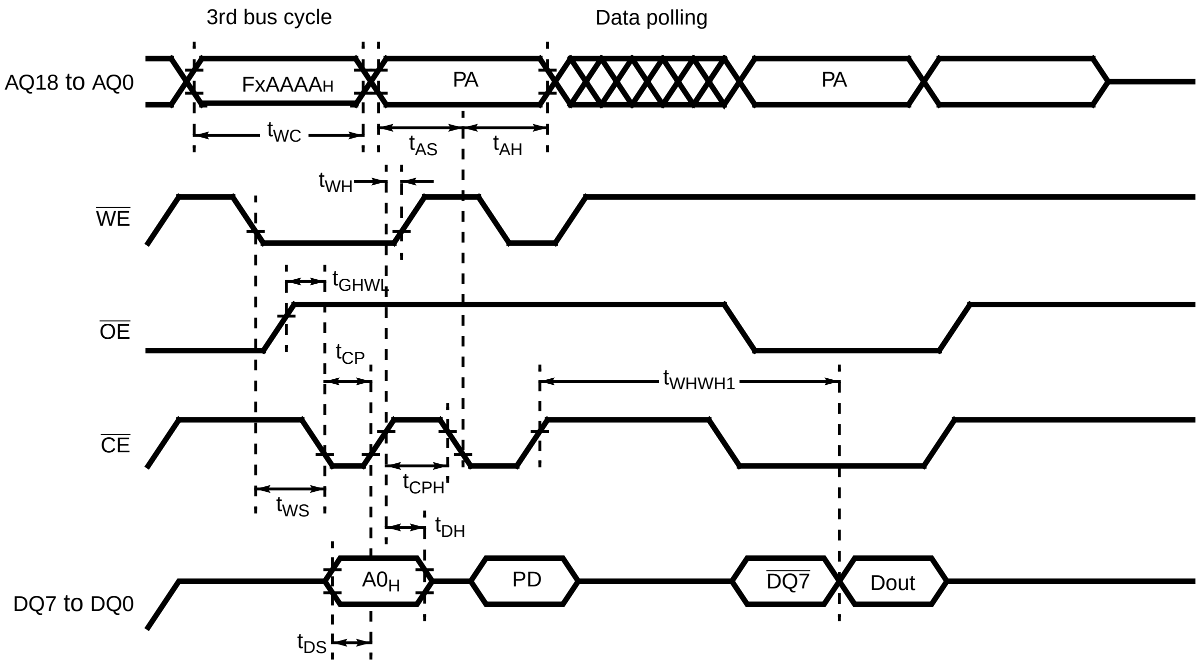
D_{OUT} : Output of write data

Note:

- Describes the last 2-bus cycle of 4-bus cycle sequences.
- "Fx" in "FxAAAA" described as address is any of FF.

■ Write, Data Polling, Read (CE control)

Figure C-3 Timing Diagram for Write Access (CE control)



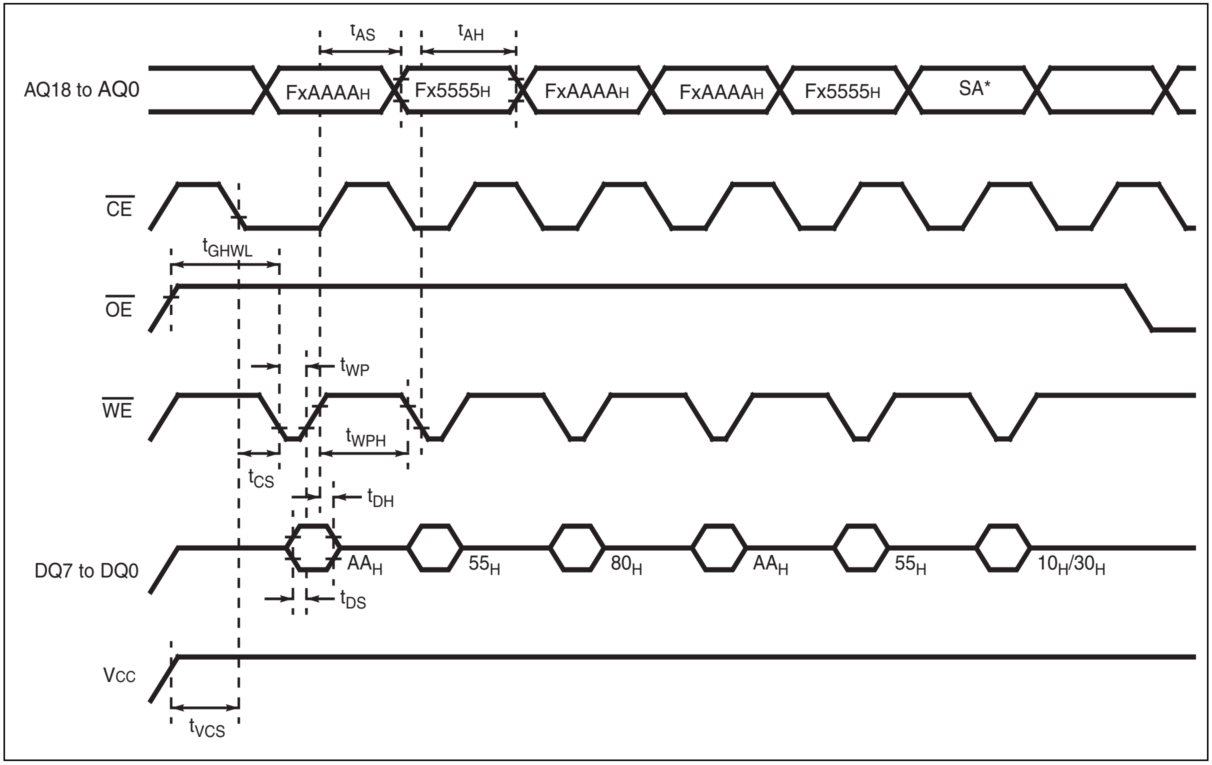
PA : Write address
PD : Write data
$\overline{DQ}_7$: Reverse output of write data
D_{OUT} : Output of write data

Note:

- Describes the last 2-bus cycle of 4-bus cycle sequences.
- "Fx" in "FxAAAA" described as address is any of F.

■ Chip Erase/sector Erase Command Sequence

Figure C-4 Timing Diagram for Write Access (chip erasing/sector erasing)

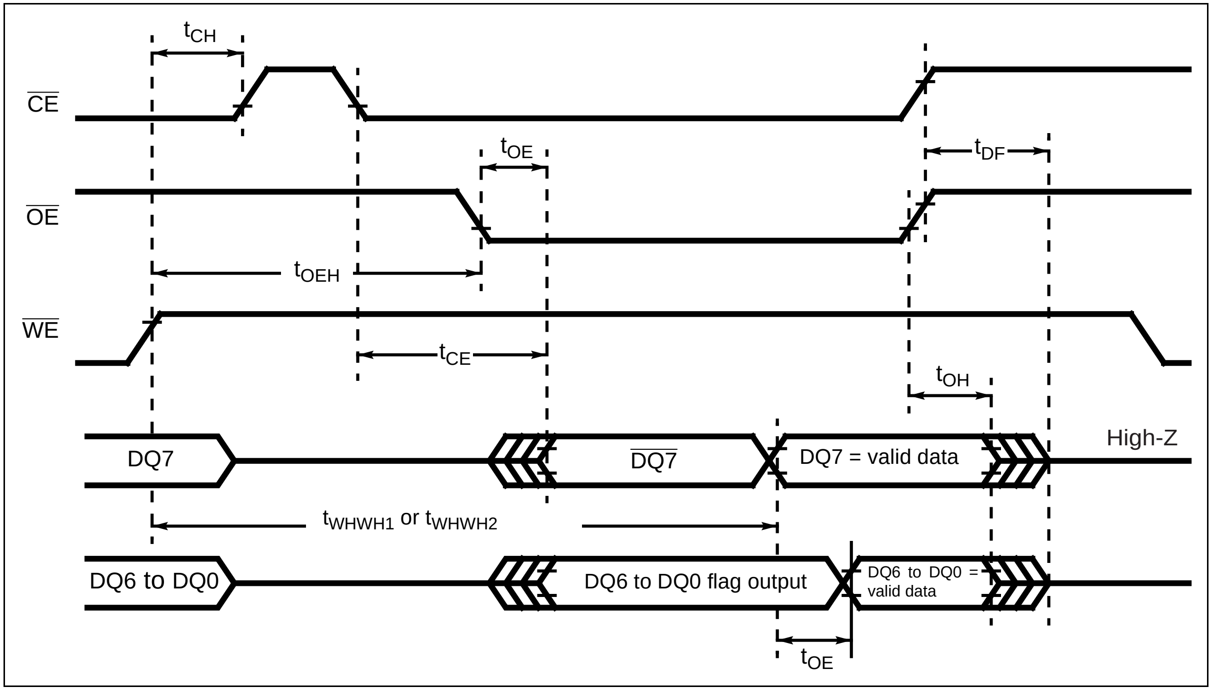


Notes:

- SA is the sector address at erasing sector.
- The address is $FxAAAA_H$ at erasing sector.
- "Fx" in "FxAAAA" described as address is any of F.

■ Data Polling

Figure C-5 Timing Diagram for Data Polling

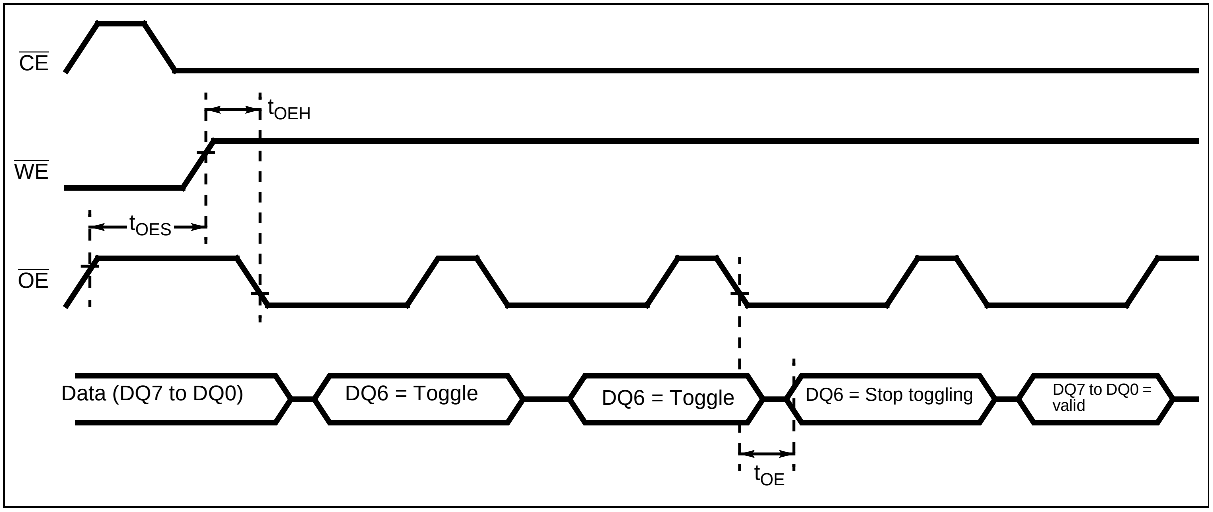


Note:

DQ7 is valid data (The device terminates automatic operation).

■ Toggle Bit

Figure C-6 Timing Diagram for Toggle Bit

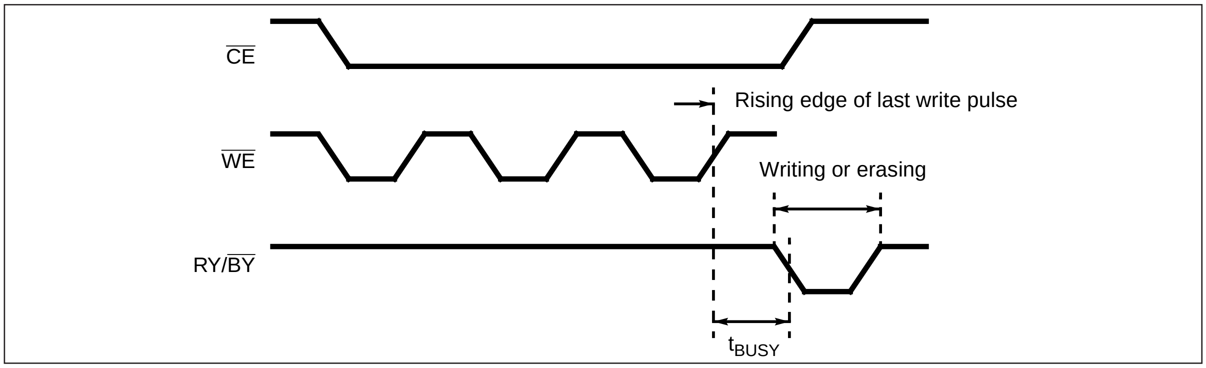


Note:

DQ6 stops toggling (The device terminates automatic operation).

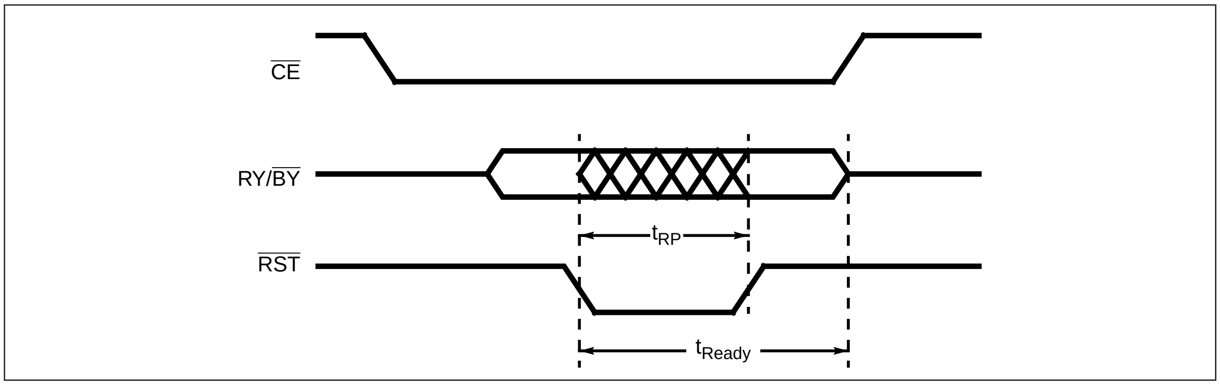
■ RY/BY Timing during Writing/erasing

Figure C-7 Timing Diagram for Output of RY/BY Signal during Writing/erasing



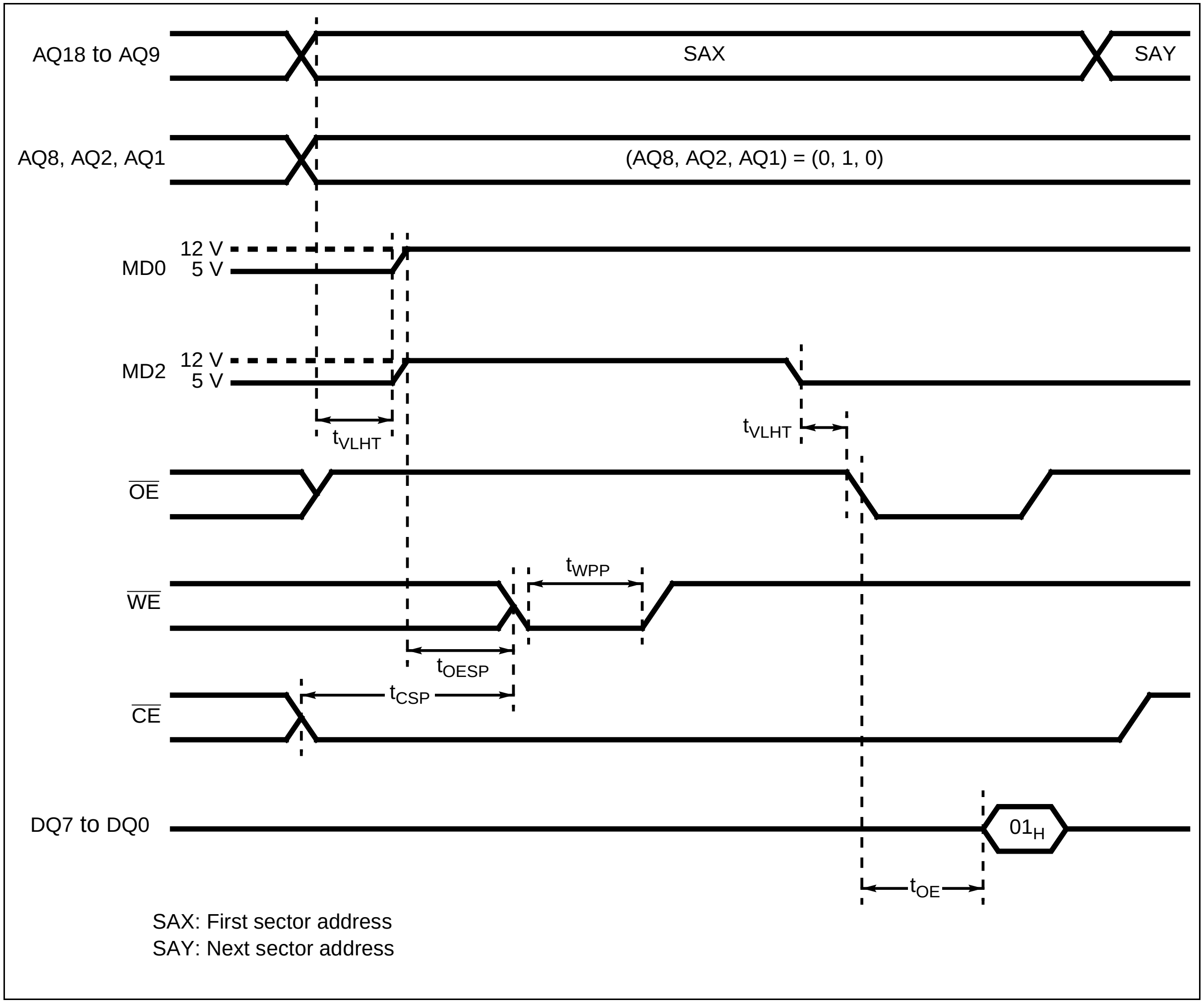
■ RST and RY/BY Timing

Figure C-8 Timing Diagram for Output of RY/BY Signal at Hardware Reset



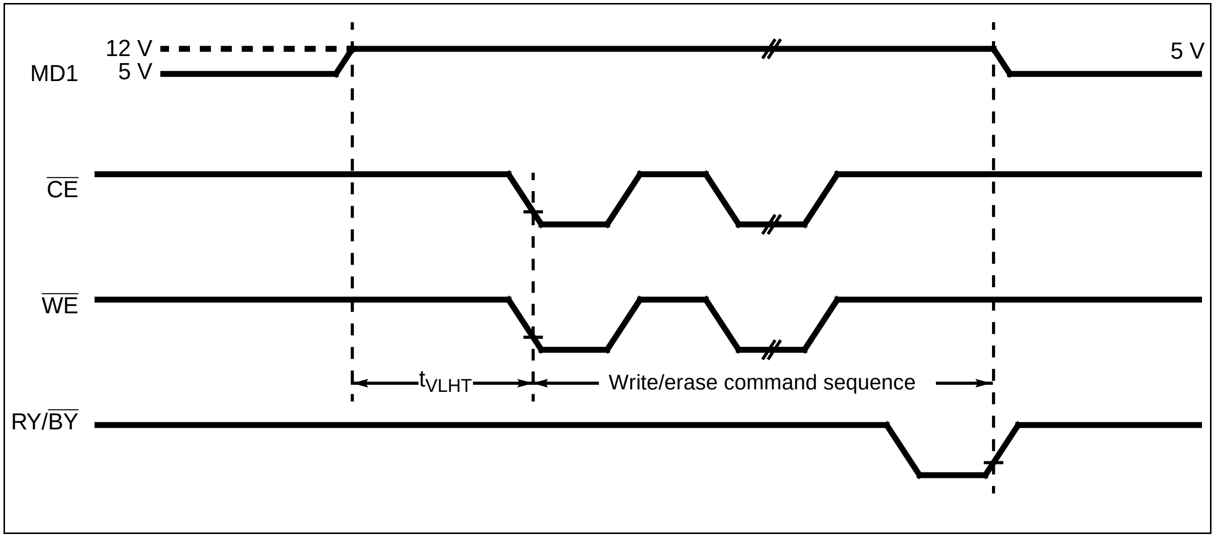
■ Enable Sector Protect/verify Sector Protect

Figure C-9 Enable Sector Protect/verify Sector Protect



■ Temporary Sector Protect Cancellation

Figure C-10 Temporary Sector Protect Cancellation



APPENDIX D List of Interrupt Vectors

The interrupt vector table to be referenced for interrupt processing is allocated to FFFC00$_H$ to FFFFFF$_H$ in the memory area and also used for software interrupts.

■ List of Interrupt Vectors

Table D-1 lists the interrupt vectors for the MB90360 series.

Table D-1 Interrupt Vectors (1/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address L	Vector address H	Vector address bank	Mode register
		Number	Address				
INT 1*	--	--	--	FFFFFC$_H$	FFFFFD$_H$	FFFFFE$_H$	Unused
INT 2*	--	--	--	FFFFF8$_H$	FFFFF9$_H$	FFFFFA$_H$	Unused
⋮	--	--	--	⋮	⋮	⋮	⋮
INT 7*	--	--	--	FFFFE0$_H$	FFFFE1$_H$	FFFFE2$_H$	Unused
INT 8	Reset	--	--	FFFFDC$_H$	FFFFDD$_H$	FFFFDE$_H$	FFFFDF$_H$
INT 9	INT9 instruction	--	--	FFFFD8$_H$	FFFFD9$_H$	FFFFDA$_H$	Unused
INT 10	Exception	--	--	FFFFD4$_H$	FFFFD5$_H$	FFFFD6$_H$	Unused
INT 11	Reserved	ICR00	0000B0$_H$	FFFFD0$_H$	FFFFD1$_H$	FFFFD2$_H$	Unused
INT 12	Reserved			FFFFCC$_H$	FFFFCD$_H$	FFFFCE$_H$	Unused
INT 13	CAN1 RX	ICR01	0000B1$_H$	FFFFC8$_H$	FFFFC9$_H$	FFFFCA$_H$	Unused
INT 14	CAN1 TX/NS			FFFFC4$_H$	FFFFC5$_H$	FFFFC6$_H$	Unused
INT 15	Reserved	ICR02	0000B2$_H$	FFFFC0$_H$	FFFFC1$_H$	FFFFC2$_H$	Unused
INT 16	Reserved			FFFFBC$_H$	FFFFBD$_H$	FFFFBE$_H$	Unused
INT 17	Reserved	ICR03	0000B3$_H$	FFFFB8$_H$	FFFFB9$_H$	FFFFBA$_H$	Unused
INT 18	Reserved			FFFFB4$_H$	FFFFB5$_H$	FFFFB6$_H$	Unused
INT 19	16-bit reloadtimer2	ICR04	0000B4$_H$	FFFFB0$_H$	FFFFB1$_H$	FFFFB2$_H$	Unused
INT 20	16-bit reloadtimer3			FFFFAC$_H$	FFFFAD$_H$	FFFFAE$_H$	Unused
INT 21	Reserved	ICR05	0000B5$_H$	FFFFA8$_H$	FFFFA9$_H$	FFFFAA$_H$	Unused
INT 22	Reserved			FFFFA4$_H$	FFFFA5$_H$	FFFFA6$_H$	Unused
INT 23	PPG C/D	ICR06	0000B6$_H$	FFFFA0$_H$	FFFFA1$_H$	FFFFA2$_H$	Unused
INT 24	PPG E/F			FFFF9C$_H$	FFFF9D$_H$	FFFF9E$_H$	Unused
INT 25	Time base timer	ICR07	0000B7$_H$	FFFF98$_H$	FFFF99$_H$	FFFF9A$_H$	Unused
INT 26	External interrupt 8 to 11			FFFF94$_H$	FFFF95$_H$	FFFF96$_H$	Unused
INT 27	Watch Timer	ICR08	0000B8$_H$	FFFF90$_H$	FFFF91$_H$	FFFF92$_H$	Unused
INT 28	External interrupt 12 to 15			FFFF8C$_H$	FFFF8D$_H$	FFFF8E$_H$	Unused
INT 29	A/D Converter	ICR09	0000B9$_H$	FFFF88$_H$	FFFF89$_H$	FFFF8A$_H$	Unused
INT 30	I/O Timer 0			FFFF84$_H$	FFFF85$_H$	FFFF86$_H$	Unused

Table D-1 Interrupt Vectors (2/2)

Interrupt request	Interrupt cause	Interrupt control register		Vector address L	Vector address H	Vector address bank	Mode register
		Number	Address				
INT 31	Reserved	ICR10	0000BA$_H$	FFFF80$_H$	FFFF81$_H$	FFFF82$_H$	Unused
INT 32	Reserved			FFFF7C$_H$	FFFF7D$_H$	FFFF7E$_H$	Unused
INT 33	Input capture 0 to 3	ICR11	0000BB$_H$	FFFF78$_H$	FFFF79$_H$	FFFF7A$_H$	Unused
INT 34	Reserved			FFFF74$_H$	FFFF75$_H$	FFFF76$_H$	Unused
INT 35	UART 0 RX	ICR12	0000BC$_H$	FFFF70$_H$	FFFF71$_H$	FFFF72$_H$	Unused
INT 36	UART 0 TX			FFFF6C$_H$	FFFF6D$_H$	FFFF6E$_H$	Unused
INT 37	UART 1 RX	ICR13	0000BD$_H$	FFFF68$_H$	FFFF69$_H$	FFFF6A$_H$	Unused
INT 38	UART 1 TX			FFFF64$_H$	FFFF65$_H$	FFFF66$_H$	Unused
INT 39	Reserved	ICR14	0000BE$_H$	FFFF60$_H$	FFFF61$_H$	FFFF62$_H$	Unused
INT 40	Reserved			FFFF5C$_H$	FFFF5D$_H$	FFFF5E$_H$	Unused
INT 41	Flash Memory	ICR15	0000BF$_H$	FFFF58$_H$	FFFF59$_H$	FFFF5A$_H$	Unused
INT 42	Delayed interrupt module			FFFF54$_H$	FFFF55$_H$	FFFF56$_H$	Unused
INT 43	--	--	--	FFFF50$_H$	FFFF51$_H$	FFFF52$_H$	Unused
⋮	--	--	--	⋮	⋮	⋮	⋮
INT 254	--	--	--	FFFC04$_H$	FFFC05$_H$	FFFC06$_H$	Unused
INT 255	--	--	--	FFFC00$_H$	FFFC01$_H$	FFFC02$_H$	Unused

*: When PCB is FF$_H$, the vector area for the CALLV instruction is the same as that for INT #vct8 (#0 to #7).
Care must be taken when using the vector for the CALLV instruction.

■ Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Table D-2 summarizes the relationships among the interrupt causes, interrupt vectors, and interrupt control registers of the MB90360 series.

Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers (1/2)

Interrupt cause	EI^{2}OS clear	DMA channel number	Interrupt vector	Interrupt control register	
			Number	ICR	Address
Reset	N	#08	FFFFDC$_H$	-	-
INT9 instruction	N	#09	FFFFD8$_H$	-	-
Exception	N	#10	FFFFD4$_H$	-	-
Reserved	N	#11	FFFFD0$_H$	ICR00	0000B0$_H$
Reserved	N	#12	FFFFCC$_H$		
CAN 1 RX	N	#13	FFFFC8$_H$	ICR01	0000B1$_H$
CAN 1 TX/NS	N	#14	FFFFC4$_H$		
Reserved	N	#15	FFFFC0$_H$	ICR02	0000B2$_H$
Reserved	N	#16	FFFFBC$_H$		
Reserved	N	#17	FFFFB8$_H$	ICR03	0000B3$_H$
Reserved	N	#18	FFFFB4$_H$		
16-bit reload timer 2	Y1	#19	FFFFB0$_H$	ICR04	0000B4$_H$
16-bit reload timer 3	Y1	#20	FFFFAC$_H$		
Reserved	N	#21	FFFFA8$_H$	ICR05	0000B5$_H$
Reserved	N	#22	FFFFA4$_H$		
PPG C/D	N	#23	FFFFA0$_H$	ICR06	0000B6$_H$
PPG E/F	N	#24	FFFF9C$_H$		
Time base timer	N	#25	FFFF98$_H$	ICR07	0000B7$_H$
External interrupt 8 to 11	Y1	#26	FFFF94$_H$		
Watch timer	N	#27	FFFF90$_H$	ICR08	0000B8$_H$
External interrupt 12 to 15	Y1	#28	FFFF8C$_H$		
A/D converter	Y1	#29	FFFF88$_H$	ICR09	0000B9$_H$
I/O timer 0	N	#30	FFFF84$_H$		
Reserved	N	#31	FFFF80$_H$	ICR10	0000BA$_H$
Reserved	N	#32	FFFF7C$_H$		
Input capture 0 to 3	Y1	#33	FFFF78$_H$	ICR11	0000BB$_H$
Reserved	N	#34	FFFF74$_H$		
UART 0 RX	Y2	#35	FFFF70$_H$	ICR12	0000BC$_H$
UART 0 TX	Y1	#36	FFFF6C$_H$		

Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers (2/2)

Interrupt cause	EI^2OS clear	DMA channel number	Interrupt vector	Interrupt control register	
			Number	ICR	Address
UART 1 RX	Y2	#37	$FFFF68_H$	ICR13	$0000BD_H$
UART 1 TX	Y1	#38	$FFFF64_H$		
Reserved	N	#39	$FFFF60_H$	ICR14	$0000BE_H$
Reserved	N	#40	$FFFF5C_H$		
Flash memory	N	#41	$FFFF58_H$	ICR15	$0000BF_H$
Delayed interrupt generation module	N	#42	$FFFF54_H$		

Y1: An EI^2OS interrupt clear signal or EI^2OS register read access clears the interrupt request flag.

Y2: An EI^2OS interrupt clear signal or EI^2OS register read access clears the interrupt request flag. A stop request is issued.

N: An EI^2OS interrupt clear signal does not clear the interrupt request flag.

Note:

For a peripheral module having two interrupt causes for one interrupt number, an EI^2OS interrupt clear signal clears both interrupt request flags.

When EI^2OS ends, an EI^2OS clear signal is sent to every interrupt flag assigned to each interrupt number.

EI^2OS is activated when one of two interrupts assigned to an interrupt control register (ICR) is caused while EI^2OS is enabled. This means that an EI^2OS descriptor that should essentially be specific to each interrupt cause is shared by two interrupts. Therefore, while one interrupt is enabled, the other interrupt must be disabled.

INDEX

The index follows on the next page.
This is listed in alphabetic order.

Index

CM44-10136-1E

FUJITSU SEMICONDUCTOR • CONTROLLER MANUAL

F^{2}MC^{TM}-16LX

16-BIT MICROCONTROLLER

MB90360 Series

HARDWARE MANUAL

April 2005 the first edition

Published **FUJITSU LIMITED** Electronic Devices
Edited Business Promotion Dept.